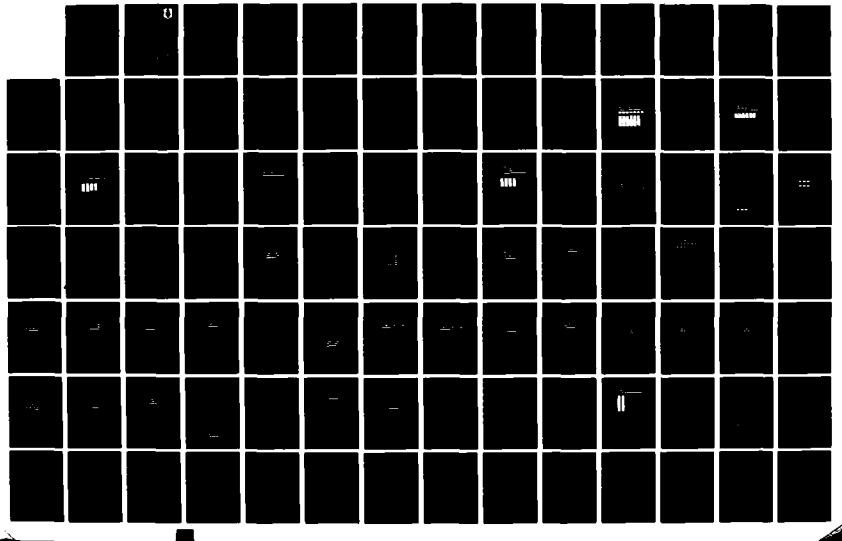
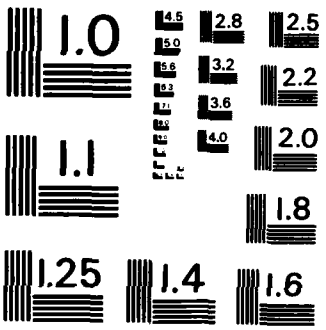
AD-A161 556 ASIM (AUTOMATED INTERACTIVE SIMULATION MODELING SYSTEM) 1/4
VAX VERSION USER'.. (U) HUGHES AIRCRAFT CO FULLERTON CA
GROUND SYSTEMS GROUP S KNEEBURG FEB 85 ESD-TR-85-127
UNCLASSIFIED F33615-81-C-5098 F/G 9/2 NL





ESD-TR-85-127

(12)

AISIM VAX VERSION USER'S MANUAL

S. KNEEBURG

Hughes Aircraft Company
Ground Systems Group
P.O. Box 3310
Fullerton, CA 92634

February 1985

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED

Prepared for

ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
DEPUTY FOR ACQUISITION LOGISTICS AND TECHNICAL OPERATIONS
HANSCOM AIR FORCE BASE, MASSACHUSETTS 01731

LEGAL NOTICE

When U.S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

OTHER NOTICES

Do not return this copy. Retain or destroy.

REVIEW AND APPROVAL

This technical report has been reviewed and is approved for publication.

N. ANN KUO, 2Lt, USAF
Project Manager,
Requirements Analysis

WILLIAM J. LETENDRE
Program Manager,
Computer Resource Management
Technology

FOR THE COMMANDER

ROBERT J. KENT
Director, Computer Systems Engineering
Deputy for Acquisition Logistics
and Technical Operations

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

AD-A161 556

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS NONE
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) ESD-TR-85-127	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION HUGHES AIRCRAFT COMPANY GROUND SYSTEMS GROUP 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION COMPUTER RESOURCE MANAGEMENT TECHNOLOGY PROGRAM, DEPUTY FOR (OVER)
6c. ADDRESS (City, State and ZIP Code) P. O. BOX 3310 FULLERTON, CA 92634	7b. ADDRESS (City, State and ZIP Code) ELECTRONIC SYSTEMS DIVISION (AFSC) HANSCOM AFB, MA 01731
8a NAME OF FUNDING/SPONSORING ORGANIZATION COMPUTER RESOURCE (OVER) 8b. OFFICE SYMBOL (If applicable) ESD/ALSE	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-81-C-5098
8c ADDRESS (City, State and ZIP Code) ELECTRONIC SYSTEMS DIVISION (AFSC) HANSCOM AFB, MA 01731	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO.: 64740F PROJECT NO.: 2522 TASK NO.: WORK UNIT NO.:
11 TITLE (Include Security Classification) AISIM VAX VERSION USER'S MANUAL (U)	
12. PERSONAL AUTHOR(S) S. KNEEBURG	
13a. TYPE OF REPORT FINAL 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 1985 FEBRUARY 15. PAGE COUNT 306
16. SUPPLEMENTARY NOTATION	
17. COSATI CODES FIELD / GROUP / SUB GR.	18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number) AISIM DESIGN PROCESSES SIMULATION MODELING ARCHITECTURE

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

THIS DOCUMENT IS THE USER'S MANUAL FOR THE AUTOMATED INTERACTIVE SIMULATION MODELING SYSTEM (AISIM).
THIS MANUAL PROVIDES THE USER WITH A COMPREHENSIVE GUIDE FOR USING THIS SYSTEM TO PERFORM HIGH LEVEL DISCRETE-EVENT SIMULATION OF COMPUTER-BASED SYSTEMS.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☒ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE NUMBER (Include Area Code) 22c. OFFICE SYMBOL

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

7a. NAME OF MONITORING ORGANIZATION (CONTINUED)
ACQUISITION LOGISTICS AND TECHNICAL OPERATIONS

8a. NAME OF FUNDING/SPONSORING ORGANIZATION (CONTINUED)
MANAGEMENT TECHNOLOGY PROGRAM, DEPUTY FOR ACQUISITION LOGISTICS AND TECHNICAL OPERATIONS

ACKNOWLEDGEMENTS

This manual was prepared by Hughes Aircraft Company under contract # F33615-81-C-5098. The work was sponsored by the Computer Engineering Applications Division (ALSE), Deputy for Acquisition Logistics and Technical Operations of the Electronic Systems Division (ESD) of the United States Air Force, Hanscom AFB, MA 01731. Funding for the effort was provided by the Air Force Computer Resource Management Technology Program, Program Element 64740F.

Program Element 64740F is the Air Force engineering development program established to develop and transfer into active use the technology, tools, and techniques needed to cope with the explosive growth in Air Force systems that use computer resources. The goals of the Program are to:

(a) Provide for the transition of computer system technology developments in laboratories, industry, and academia to Air Force systems;

(b) Develop and apply software acquisition management techniques to reduce life cycle costs;

(c) Provide improved software design tools;

(d) Address the various problems associated with computer security;

(e) Develop advanced software engineering tools, techniques, and systems;

(f) Support the implementation of high order languages, e.g. Ada;

(g) Address human engineering for computer systems; and

(h) Develop and apply computer simulation techniques for the acquisition process.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

SECTION 1

INTRODUCTION

1.1 PURPOSE AND SCOPE

The Automated Interactive Simulation Modeling System (AISIM) provides the user with the ability to do high level simulation of complex operational and distributed data processing systems. The purpose of this manual is to provide the AISIM user with a comprehensive guide for the use of the AISIM system on a VAX 11 780 computer.

1.2 ORGANIZATION

This manual is ~~organized to provide~~ a straightforward reference document for the AISIM user. Chapter 1 introduces this document, detailing the organization of this document, the document conventions and applicable documents Chapter 2 is an overview of the concepts used in modeling and simulation of systems using AISIM. Chapter 3 contains a detailed description of the AISIM modeling constructs. Chapter 4 describes the interface between the AISIM software and the host computer's time sharing system. Chapters 5 through 10 present information of the various system user interface levels, including detailed descriptions of prompts and commands. Chapter 11 discusses AISIM simulation results and how to interpret them. Appendix A presents operational procedures and other information which is useful for the user to know but not mandatory for using the system. Appendix B lists simulation error messages with a description of their meaning. Appendix C is a glossary of AISIM terms. Appendix D contains a detailed description of the message routing submodel described in section 3.

1.3 DOCUMENTATION CONVENTIONS

The descriptions of AISIM commands given in this manual use the following notations to define the syntax and format of the AISIM commands:

1. Commands shown in the format below are equivalent:

   DESIGN

   D

   The latter is an abbreviation for the former.

2. Required parameters are enclosed in braces:

   {language}

3. Optional parameters are enclosed in brackets:

   [NOXLATE]

   Default values exist for all optional parameters.

4. The brace and bracket symbols are used only to define the format. They should never be typed in the actual command statement.

   braces { }

   brackets [ ]

5. The symbols listed below should be typed in a command statement exactly as shown in the command statement definition.

   apostrophe '

   comma ,

   parentheses ( )

   period .

6. Words in lower case appearing in a command definition represent variables for which the user should substitute specific information in the actual command.

   EXAMPLE: If "database" appears in a command definition, the user should substitute a specific name of a database (for example, CONTACT) for the variable when the command is entered on the terminal.

7. All upper case words and letters in a command definition, such as a command name or a parameter name, must be typed as part of the command statement.

8. All command names and associated parameters must be separated from each other by the appropriate delimiter, as shown in the command definition. Delimiters are either a comma or a blank depending on the context. A blank is entered on the terminal by pressing the space bar at the bottom of the terminal keyboard.

   EXAMPLE: BACKUP [PROJECT(database)]

   If the optional parameter is used, it must be separated from the command name BACKUP by a blank ( ), i.e.,

   BACKUP PROJECT(contact)

When a comma is to be used as the delimiter, it will be specified as part of the command definition.

EXAMPLE: DEFPLOT {entity-type},{entity-name}

In this example the command name DEFPLOT would be separated from the required parameter {entity-type} by a blank and the two required parameters would be separated from each other by a comma, i.e.,

DEFPLOT R,resource

9. The references in this document to specific words which are AISIM entities, will appear with an initial capital. This is to distinguish the reference to an AISIM specific concept from a common interpretation of the word.

EXAMPLE: Process - Occurrences of this refer to the AISIM entity.

1.4 APPLICABLE DOCUMENTS

The following documents provide additional information relevant to the operation and use of AISIM:

AISIM Training Manual

AISIM Training Examples Manual

SECTION 2

AISIM CONCEPTS

The Automated Interactive Simulation Modeling System (AISIM) provides a tool for the analysis of complex systems. The tool is designed for the operations analyst or engineer as a workbench for investigating the impact of system alternatives. AISIM provides a graphics language for the expression of systems, a database for storing a system's design and a simulation capability for analyzing the system. AISIM is applicable to design analysis of hypothetical systems and to the operations analysis of existing systems.

AISIM is a computer program that allows for the simulation of complex systems by a user without the need for the user to do additional programming. The program can be executed interactively by a user communicating with a host computer through a terminal. By using the host computer in an interactive mode, an AISIM user can use AISIM to obtain timely data to support decisions on how a system is to function.

2.1 CHARACTERISTICS OF SYSTEMS MODELED BY AISIM

AISIM supports the design and analysis of systems having any of the following characteristics.

1. Procedural operations -- Processes in the system can be described by a sequence of steps that describe the logic of every operation (e.g., operator actions, operating system logic, applications logic, man-machine interface, real time input processing).

2. Parallel Processing -- Any number of processes can occur simultaneously.

3. Shared Resources -- Some processes require resources that are contended for by other processes (e.g., two I/O requests contending for a single channel). Queueing is reflected in the degradation of the time required to complete processes suffering resource contention (e.g., large queues behind bottlenecks in a network).

4. Operational loading -- The operation of the system is a function both of its internal structure and of the environmental pressures on it.

5. Process communication -- Processes transfer data and materials to other processes in the system (e.g., both message routing and network control information communication can be easily represented).

6. Interconnected network -- Network architectures consisting of interconnected nodes can be represented in AISIM. System constructs allow the user to define the routing of messages through the described architecture. AISIM also allows for the modeling of systems abstracted from any particular architecture.

These characteristics are generic to a large class of systems including military, computer, and industrial systems.

2.2 MODELING

In scientific and engineering usage, a model is a simplified (or idealized) representation of a system that is advanced as a basis for calculations, predictions or further investigation. AISIM modeling fits comfortably under this general characterization, but AISIM is especially useful for the modeling of systems which incorporate parallel processing (simultaneous activity) and networks. AISIM is particularly suited to the modeling of embedded computer systems for command, control and communication applications.

There are many applications of simulation modeling in this problem area. AISIM models are representative, discrete event simulation models used for predictive operations analysis. What this means is that entities in a real system are mapped onto AISIM entities which have a very close functional relationship. AISIM entities respond to simulated conditions much like the real entities do under actual conditions. This is in contrast to functional modeling where the real system is described in terms of equations in differential calculus. The emphasis in representative modeling is on describing the system.

Generally, determining and clearly describing the system is the first major obstacle a modeler must confront. If a system is in the design phase, then no data is available on how it will perform or what the major bottlenecks will be. For existing systems these characteristics may be known but the combination of events that cause problems may not be understood. In both cases, much can be learned from modeling the system.

A key concept to keep in mind is that models are a simplified description of a system. This implies that some elements of the real system may not be represented in the model. The challenge in modeling is to represent all the elements of critical interest to the system dynamics in the model. This requires some thought to the development of the model.

2.3 DESIGNING MODELS

A model should be carefully designed before being built. The key activities addressed during the design phase are the following:

1. Understand the Model and Collect Relevant Data -- To model any system effectively, a modeler has to know something about the system. Building an executable simulation model requires that the system have an accurate and sufficiently detailed description. A modeler must be

aware of the functions performed in a system which effect the dynamics of the operation. A modeler must also know the characteristics of all the elements that perform work, create data, control processing, interrupt normal operations and produce output. This data can be obtained from design specifications, hardware specifications, previous studies or empirical testing. It is important to collect good data because that data becomes the foundation of the model.

2. Determine Model Boundaries -- Systems modeled by AISIM generally consist of many subsystems. The problems caused by the combination of subsystem activities are of interest to the analyst. AISIM provides varying levels of detail in modeling a subsystem. Sometimes the activity can be viewed as a black box. The flow of control through this box can simply be represented by a delay. This type of phenomena is modeled by AISIM with the Action entity. Other times, the characteristics of a subsystem can be represented by a mathematical function. AISIM has such a functional capability with the EVAL Primitive and Table entity. If an activity is more complicated, it can be described by logic. In this case, AISIM allows the modeler to go to his own level of detail by building a Process. Setting the boundaries of an AISIM model is precisely what the modeler does in deciding which of these constructs will be used to model the elements of a system. A method of paper modeling developed for software design is known as "structured design". This method uses structure charts, hierarchical charts showing calling sequences, to describe functional processing. This method has been used successfully with AISIM. An alternate method would be to create flow charts of the various system functions.

3. Determine Experimental Method -- A model allows an analyst to run experiments on a system to predict how an operation will behave. Before any effort is expended in building a model, the output of simulation runs must be considered. Monitors can be designed to provide data on the system's operation. Experiments can be designed to validate the model.

2.4 CONSTRUCTING AN AISIM MODEL

2.4.1 Charting a Paper Model

In building a model, a modeler maps the elements of a system onto the constructs of the simulation language. To do this, the modeler must be familiar with the characteristics and relationships of both the simulation tool and the real-world system. The mapping is not always clear-cut and usually requires iteration. The modeler charts out what processing takes place in a system, where resources are allocated, how processes communicate and where activities initiate. This chart is referred to as a paper model. It may be derived from an understanding of the system's functions and a graphical representation of its network. On the paper model, the modeler names the entities in the system that will be modeled by AISIM entities - Processes, Resources, Items, Queues, Tables, etc.

2.4.2 Defining the AISIM Model

An AISIM model is built by defining AISIM entities to represent system entities. This is done interactively on the computer. AISIM solicits relevant data for defining all design entities.

2.5 AISIM MODELING ENTITIES

As mentioned earlier, a model is a description or abstraction of a real or proposed system. To build a model with the intention of simulating its operation, we must describe the model in terms which can be interpreted, and operated upon, by the simulation system. That is, a system can be modeled using a prose description; but unless it has some systematic relation to a computer language, it would be useless as a computer model because prose is ambiguous. AISIM uses a special set of terms to describe system structure and operation called AISIM entities. A modeler must understand the meaning and use of these entities to build successful models. These entities are briefly discussed below. A detailed discussion of each of these entities is provided in section 3. Figure 2-1 also provides further insights to the meaning, use, and relationships between entities and other modeling constructs.

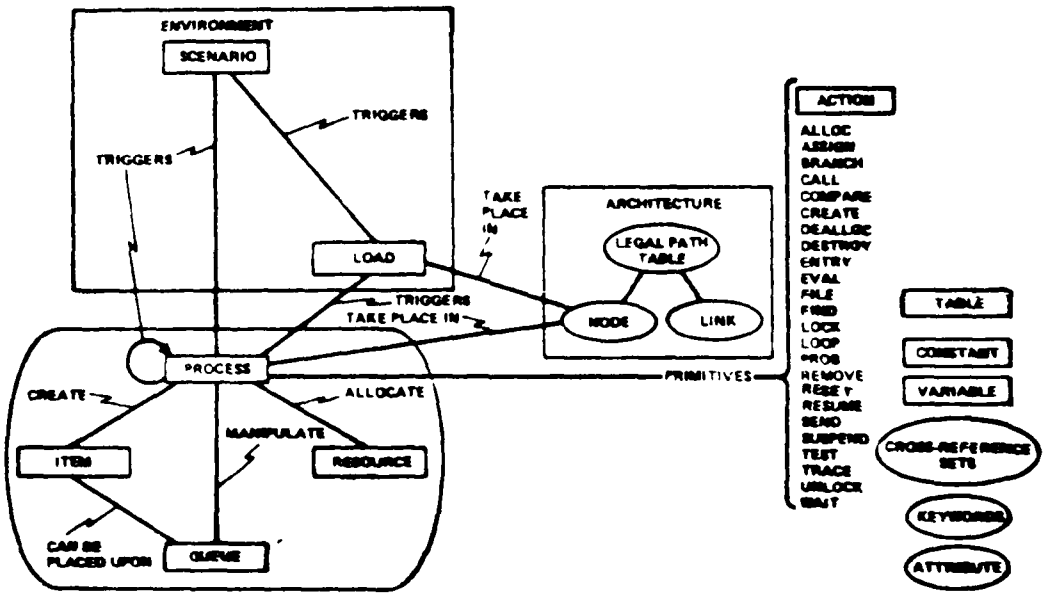


Figure 2-1. AISIM Entity Relationships

Constant - A Constant is a term whose value does not change during a simulation exercise of a model. Constants are used to represent parameters that do not vary with time or in response to the workings of the system being modeled.

Item - An Item is a transient data element and is used to represent messages (or materials or even physical objects) flowing through the system.

Load - A Load is used to represent aspects of the world outside the system that trigger the initiation of Processes. Loads represent the normal burden, i.e., occasional Process triggering, on a system.

Primitive - Primitives are logical constructs that represent steps in the modeled system's operation. There are 25 different Primitives each representing a different logical function. A sequence of Primitives compose a Process. All Primitives are listed below. The ACTION primitive has an Action entity associated with it. The Action entity is defined below.

ACTION - (See below)
ALLOC
ASSIGN
BRANCH
CALL
COMPARE
CREATE
DEALLOC
DESTROY
ENTRY
EVAL
FILE
FIND
LOCK
LOOP
PROB
REMOVE
RESET
RESUME
SEND
SUSPEND
TEST
TRACE
UNLOCK
WAIT

Process - A Process is a logical description (using Primitives) of some or all of the operations, decisions or activities of the system being modeled.

Action - An Action, which is associated with the ACTION Primitive, is used to represent the consumption of time for any action, activity, decision, etc., that consumes time. The ACTION Primitive is the only one that updates the simulation clock.

Queue - The Queue entity is used to model an ordered holding area for one or more Items. A Queue may be used to model, for example, a job queue or a memory buffer. A Queue may be defined with a maximum size parameter to model, for example, such limits as the maximum number of messages that a buffer can hold before it is overloaded. Queues bear a default size of infinite.

Resource - The Resource entity is used to model the mechanisms (people, CPU, communication lines, etc.) necessary to complete a Process. Resources generally have the property of being shared among Processes. Performance of a Process can be degraded due to contention for Resources.

Scenario - The Scenario entity is used to model the various environments in which a system must perform. A Scenario specifies the number of periods of a simulation run as well as their length (which is uniform). The Scenario schedules the initiation of Loads. It can also schedule the initiation of Processes.

Table - A Table is a user-definable function with up to fifteen pairs of data points. Tables may be defined as either continuous, discrete or alpha. A continuous Table interpolates linearly between numeric points. A discrete Table is a step function connecting numeric points. Alpha Tables are used for structuring data over non-numeric ranges and domains.

Variable - A Variable is a term whose value can change during a simulation run, either by setting it equal to a mathematical expression or through reassignment by the user between stages of a simulation.

Keywords - The keywords are system-defined variables which provide the user with information about the current state of the simulation.

SECTION 3

AISIM ENTITIES AND OTHER MODELING CONSTRUCTS

In this section AISIM's entities and other modeling constructs are described in detail. For each entity, the parameters required to define the entity and the means by which this data is requested from the user are described. Included is mention of relations between the various AISIM entities, where such mention is deemed helpful.

3.1 SCENARIO

The Scenario entity is used to represent the various environments in which the system being modeled must perform. Together with the Load entity it represents the external stimuli on a modeled system.

In a Scenario, the user defines a collection of Loads and/or Processes, together with schedule time and triggering priority for each. The Scenario calls for the initiation of activity over time by activating a Process or Load at the corresponding scheduled time.

Scenarios are divided into periods whose length and number are chosen by the user. These periods provide break points at which the user can stop a simulation to alter a variable or inspect the results up to that point. There may be up to 14 periods in a given Scenario. The form for the Scenario is shown in figure 3-1.

Figure 3-1. Form for the Scenario Entity

Following is a description of the fields in the Scenario form:

SCENARIO: Scenario name (1 to 8 characters)

PERIOD LENGTH: Amount of time in each simulated period.

DESCRIPTION: Any user comment (0 to 53 characters)

PERIODS: Mnemonic names can be entered in these fields consisting of up to 8 characters per name. The number of fields containing characters determines the number of

periods in a simulation, i.e., for each of the 14 fields in which an entry is made a period is added to the total simulation run. A Scenario can have a maximum of 14 periods.

TRIGGER: 1 to 20 Process names or Load names; each Process or Load named causes the initiation of that Process or Load at the scheduled time.

SCH TIME: The simulation time, from the start of the simulation, at which the the Load or Process specified is to be initiated.

PRIORITY: The priority the triggered Process is to have. Priority is inverse, priority 1 preempts priority 2. If a Load name is entered in the trigger field, the corresponding priority field is ignored.

Note: Constants may be used to define PERIOD LENGTH, SCH TIME, and PRIORITY.

Operation - A model database may contain more than one Scenario. However, only one can be used in any simulation. The Scenario specified will define simulation period length, and Loads and Processes to be triggered by the Scenario. The total simulation time is the product of the number of periods and the period length. The number of periods also effects the collection of plot data points. (see appendix A.3)

Scenario entities are entered using the Design User Interface EDIT comand (see section 6.1.4).

3.2 LOAD

The Load entity is used with the Scenario entity to periodically trigger Processes at specific nodes in the architecture. The Load describes which Processes will be initiated and at which nodes. An instance of the Load is triggered simultaneously at each of the specified nodes. This entity can be described in the following way: for each Process in the Load, initiate up to the maximum number at an interval determined by the schedule method, and initiate them at each of the specified nodes. The form for the LOAD entity is shown in figure 3-2.

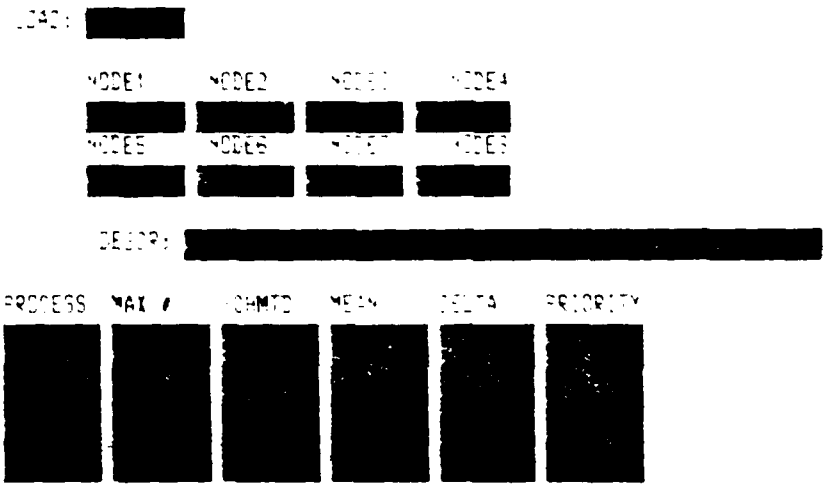


Figure 3-2. Form for the Load Entity

Following is a description of the fields in the Load form.

LOAD: Load name (1 to 8 characters)

NODES: If an architecture is used, one to eight nodes in which the Processes specified will take place. Otherwise leave blank.

DESCR: Any user comment (0 to 53 characters)

PROCESS: 1 to 5 names of Processes which the Load triggers according to schedule.

MAX #: Maximum number of times this Process is to be triggered in each execution of the Load.

SCHMTD: Statistical function to be used to determine the time between Process triggerings. It can be any of those described under SCHEDULE METHOD (see below).

MEAN: Depending upon schedule method, MEAN is used to determine the interval between each triggering of a Process. In general this is the mean inter-arrival time.

DELTA: Depending upon schedule method, DELTA is used to determine the deviation about the mean for the interval between triggerings of a Process.

PRIORITY: Priority with which the Process is to be executed. Priority is inverse, priority 1 preempts priority 2. Priority is used to determine which Process will be allowed to allocate a Resource when it is contended for by two or more Processes (see ALLOC Primitive, section 3.9.2).

SCHEDULE METHODS:

START - MEAN: inapplicable; i.e., leave field blank
DELTA: inapplicable; i.e., leave field blank

All Processes up to the maximum number are initiated at the same clock time, the start of the Load. This can be used to simulate pre-loading.

INTERVAL - MEAN: time between initiations
DELTA: inapplicable; i.e., leave field blank

One Process is initiated at every interval as defined by MEAN. The first starts at the time given by MEAN with respect to the starting time of the Load.

POISSON - MAX #: mean number in a PERIOD
MEAN: inapplicable; i.e., leave field blank
DELTA: inapplicable; i.e., leave field blank

Processes are scheduled randomly by a Poisson process. The time between Process triggerings is exponentially distributed. The MAX # parameter defines the mean number for a PERIOD. PERIOD length is defined in the Scenario.

EXPONENT - MEAN: mean time between Process triggerings
DELTA: inapplicable; i.e., leave field blank

The time passing between Process triggerings is exponentially distributed.

LOGNORML - MEAN: mean time between Process triggerings
DELTA: standard deviation of time between Process triggerings

The time passing between Process triggerings is lognormally distributed.

NORMAL - MEAN: mean time between Process triggerings
DELTA: standard deviation of time between Process triggerings

The time passing between Process triggerings is normally distributed.

UNIFORM - MEAN: mean time between Process triggerings
DELTA: range about the MEAN

The time passing between Process triggerings is uniformly distributed. The DELTA parameter specifies the difference between the largest possible time between Process triggerings and the MEAN time.

ERLANG - MEAN: mean time between Process triggerings
DELTA: order of the distribution function

The time passing between Process triggerings is Erlang distributed. The order "k" is given by the DELTA.

WEIBULL - MEAN: scale parameter.
DELTA: shape parameter

The time passing between Process triggerings is Weibull distributed.

GAMMA - MEAN: mean time between Process triggerings
DELTA: k

The time passing between Process triggerings is gamma distributed.

Operation - a Load specifies a cluster of Processes to be triggered according to a scheduling method and a priority.

Relationships - Loads are part of Scenarios and specify Processes to be triggered and nodes in which they are to be triggered.

Load entities are entered using the Design User Interface EDIT command (see section 6.1.4).

3.3 ITEM

The Item entity is used to model transient data elements that "flow" through a system. These data items, which, by the nature of their varying attribute values, permit data dependent decision making and timing.

Items can be originated, terminated and passed through the system from one Process to another through the Primitives CREATE, DESTROY, CALL and SEND. Items can also be placed on and removed from Queues via the Primitives FILE and REMOVE, and pointed to via the Primitive FIND. The form for the Item entity is shown in figure 3-3.

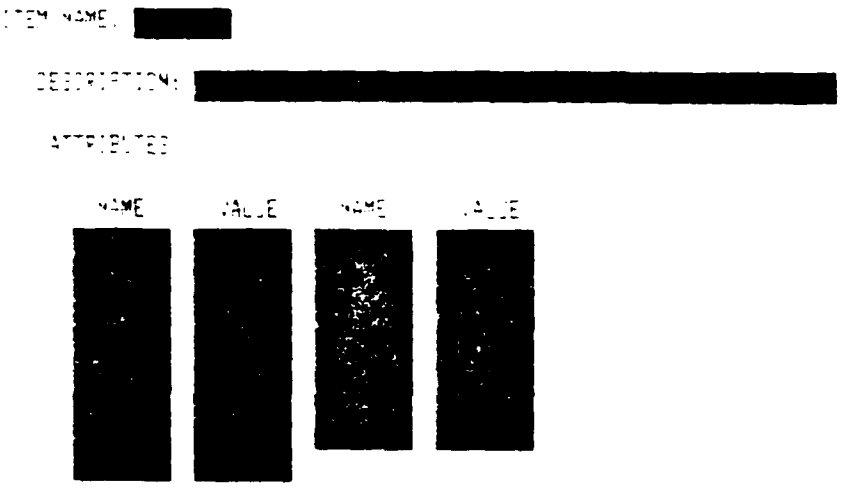


Figure 3-3. Form for the Item Entity

Following is a description of the fields in the Item form.

ITEM NAME:	1 to 8 character name of Item
DESCRIPTION:	Any user comment (0 to 53 characters)
NAME:	1 to 8 character name of an attribute of Item. An Item can have up to 15 user-defined attributes.
VALUE:	The initial value to be assigned to the corresponding attribute (integer, decimal, or character); if a name, it must be a defined Process, Resource, global Variable, Constant, Item, Queue, Table, Action, keyword or alpha literal.

NOTE: All Items have two implicitly defined attributes, TAIL and PRIORITY. TAIL is the number of the Item created, and PRIORITY is the priority of the Process that created the Item. The TAIL attribute can be used for Item matching (see SEND Primitive).

Operation - An Item is created for each occurrence of the following:

a. a CREATE Primitive that is executed - used to model transient data elements

b. a SEND Primitive that is executed in a Process which does not have an Item of the specific name attached at the time.

An Item is terminated only when the DESTROY Primitive is executed.

Attribute values are assigned at the time of creation.

Relationship - Item attributes are used by Process Primitives and attribute values can be modified by the ASSIGN Primitive.

Item entities are entered using the Design User Interface EDIT command (see section 6.1.4).

3.4 USER DEFINED QUEUES

A Queue is a global entity used to represent an ordered holding area for Items.

When a Queue is defined, a maximum size parameter is specified (the default is "infinite"). This allows Queues to model finite storage devices that have a limited capacity (e.g., a storage bin, a computer job scheduler). Once the value is defined, it may not be changed and thus this parameter must be either a numeric value or a defined Constant.

Queues are manipulated by Processes through the use of the FILE, FIND, and REMOVE Primitives. An Item may be placed on a Queue, if space exists, by using the FILE Primitive, specifying one of four location parameters: FIRST, LAST, BEFORE and NEXT. The former two parameters denote the end points of a Queue, the first and last slots. The latter two are location parameters relative to a Queue pointer (see below). If no space exists on the Queue, the Process which is executing the FILE Primitive is suspended. This condition is known as Queue blocked. In this state the Process waits until space becomes available on the Queue. Waiting for space on a Queue is by a first come first served discipline.

An Item may be taken off the Queue by using the REMOVE Primitive and specifying a location parameter (i.e., FIRST, LAST, or NEXT, where NEXT means the current Item pointed to by the Queue pointer). After an Item is removed from a Queue, it may be sent, destroyed, or otherwise modified.

An Item may not be modified, sent, or destroyed while it is on a Queue. The same Item instance may not exist on more than one Queue. Multiple Processes may access the same Queue.

A Queue pointer is maintained for each Process which references a Queue. This pointer contains the address of the entity that the Process is addressing in a Queue. The contents of the Queue pointer is determined by rules described below and in the sections on the Primitives FILE (section 3.9.12), FIND (section 3.9.13) and REMOVE (section 3.9.17):

1. The pointer contains the address of the last entity found with a FIND Primitive; otherwise,

2. The pointer contains the address of the last entity filed with a FILE Primitive; otherwise,

3. The pointer contains the address of the successor of the last entity removed with a REMOVE Primitive with a NEXT option.

The REMOVE and FIND Primitives access a Queue and set the value of the local variable referenced in the Primitive. This means that when a FIND or REMOVE Primitive is executed, the value of the local variable could be set to 0. This occurs under the following circumstances:

1. A REMOVE Primitive attempts to remove an entity from an empty Queue.

2. A FIND Primitive accesses an empty Queue.

3. The NEXT or BEFORE Item in a Queue does not exist.

The form for the Queue entity is shown in figure 3-4.

Figure 3-4. Form for the Queue Entity

Following is a description of the fields in the Queue form.

QUEUE: 1 to 8 character name of Queue

SIZE: An integer value of 1 to 8 digits, a defined Constant entity, or the word INFINITE

DESCR: Any user comment (0 to 53 characters)

Relationships - Queues are used to hold Items. Queues are manipulated by the FILE, FIND, and REMOVE Primitives.

Queue entities are entered using the Design User Interface EDIT command (see section 6.1.4). Attributes associated with Queue entities are described in section 3.12.

See section 3.5 for a description of system defined queues.

3.5 SYSTEM DEFINED QUEUES

3.5.1 States Associated with Resources

Associated with each Resource entity are four simulation states upon which statistics are kept. Three of these states apply to Resource units and one of the states applies to Processes. Resource units can be in one of the three states idle, busy, and inactive. If a Process is waiting for a Resource unit which is unavailable, the Process is in the wait state. Resource units which are idle or inactive are accounted for by counters associated with the Resource. Busy Resource units are kept on a system-defined queue called the busy queue, and Process which are waiting for Resource units are kept on a wait queue. Resources and Processes are placed in these states during the simulation as follows:

Resource units are idle while they are unallocated and available to Processes. Resource units are in the idle state: (1) at the initialization of the simulation, (2) when removed from the inactive state (by the RESET Primitive) or (3) when removed from the busy queue (by the DEALLOC Primitive).

Resource units are placed on the busy queue while they are allocated by some Process through the ALLOC Primitive. They may be removed from the busy queue (1) by being deallocated with the DEALLOC Primitive or (2) by being set inactive by the RESET Primitive.

Resource units are in the inactive state when they are not available to be allocated by Processes. Resources may be placed in this state (1) at the initialization of the simulation, (2) from the idle state by means of the RESET Primitive, and (3) from the busy state by means of the RESET Primitive.

The wait queue holds Processes that are suspended for lack of an available unit of the needed Resource. A Process is placed on this queue when either (1) it attempts to allocate the Resource (with the ALLOC Primitive) that is held by another Process of equal or "higher" priority or (2) it loses a Resource to a "higher" priority Process.

The relation between these states is illustrated in figure 3-5.

During a simulation run statistics are kept on the activity of these states. These results are presented in the simulation's Resource report. The user can access the number of Resource units or Processes currently in each of the states using attributes described in section 3.12.

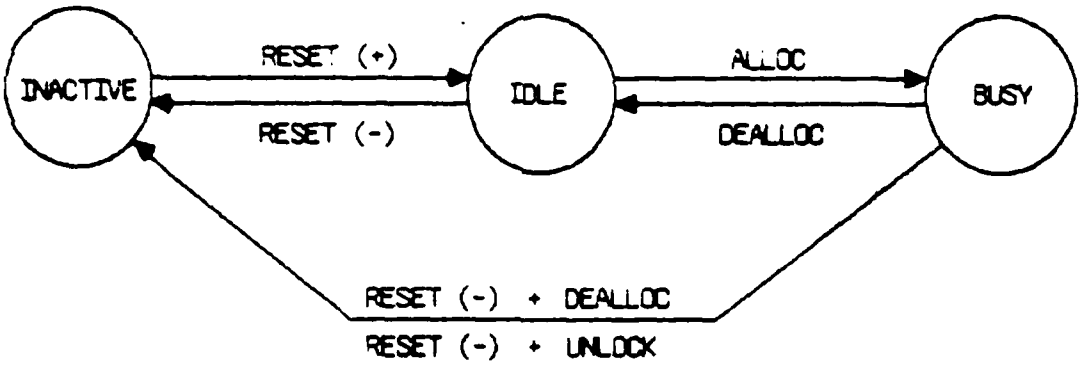


Figure 3-5. Resource States

3.5.2 Cross Reference Sets

In addition to the queues associated with Resource contention, there are eight system defined queues called "cross-reference sets". These queues correspond to the sets of names of the following AISIM entities:

1. Resource names
2. Queue names
3. Process names
4. Item names
5. Action names
6. Table names
7. Constant names
8. Variable names

What this means is that an AISIM modeler can write Processes which perform some function on each entity defined in one of the above sets.

The FIND Primitive accesses the set of names of an entity type by specifying the name, e.g., Resource, Item, Process, as the Queue field reference in the Primitive.

3.6 RESOURCE

The Resource entity is used to model the mechanisms required to perform a Process. "Mechanisms" in this context can be computer processors, memory, communications channels, support personnel, documents, etc.

Queueing for a Resource occurs only within a Process and, in particular, only where an ALLOC Primitive is used. In other words, if no ALLOC Primitive is used there will be no queueing. If no Resource is used (allocated) within a Process, the Process can be executed in parallel (simultaneously) by any number of concurrent requests and the model will represent only time delays associated with the ACTION Primitive.

When a Resource is used (allocated) by a Process, there can be only as many concurrent executions of the Process as there are Resource units available. For example, if the capacity of a Resource is one, then any Processes which allocate that Resource will be executed serially (one at a time). Execution concurrency is controlled only between the allocation and deallocation of the Resource (i.e., if the ALLOC Primitive is the second Primitive in a Process, the first Primitive can be executed concurrently by any number of requests whereas the ALLOC Primitive can be executed concurrently by only as many requests as the Resource has units available).

If no Resource units are available (i.e., idle or presently allocated to a lower priority Process) when an ALLOC Primitive is attempted, the Process' allocation request is merged onto a wait queue associated with the Resource. How the request is merged depends on the priority of the request. The request is merged and sorted by inverse priority (priority 1 preempts priority 2). Within priority the sorting is done first-in-first-out. When deallocation of the Resource (by some other Process) has resulted in enough units to satisfy the requests, and the request has moved to the top of the wait queue, then the request is removed from the queue, the allocation is performed, and the Process is executed. Note that a deallocation of several units may result in several requests being removed from the queue simultaneously. For allocation requests of multiple units, the user can specify whether the units are to be allocated as they become available or only when all units are available at the same time.

If, when the ALLOC Primitive is attempted, there is a lower priority Process possessing the desired Resource units, then the higher priority Process will "steal" those units. The lower priority Process will be suspended while it waits for Resource units. It will be placed on the wait queue but its seniority is based upon the time of its first allocation attempt, not the time it lost its Resources.

The Resource entity provides the most interesting and useful simulation results; e.g., delays, bottlenecks, utilization percentages, and traffic statistics. Therefore, the use of Resources should be carefully designed from both the standpoint of model credibility and the specification of required simulation output.

The form for the Resource entity is shown in figure 3-6.

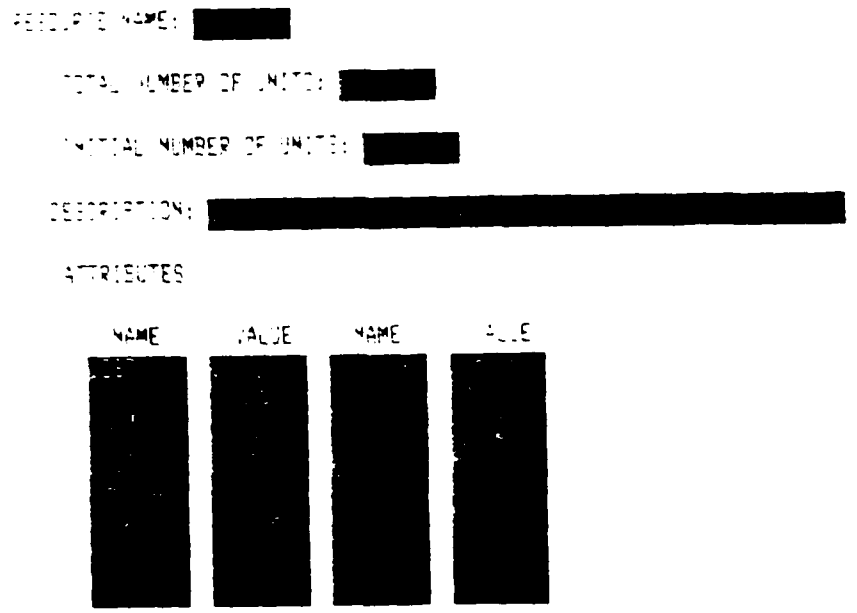


Figure 3-6. Form for the Resource Entity

Following is a description of the fields in the Resource form:

RESOURCE NAME: 1 to 8 character name of Resource

TOTAL NUMBER: Maximum number of units of the Resource that can be allocated (integer or named Constant).

INITIAL NUMBER: Number of units available for allocation at the start of the simulation (integer or named Constant).

DESCRIPTION: Any user comment (0 to 53 characters)

NAME: 1 to 8 character name of user defined attribute
Cost is a default attribute to document the cost of the Resource.

VALUE: Initial value to be assigned to an attribute; can be single precision real or integer number, or the name of a defined Variable, Constant, Process, Item, Resource, Queue, Action, Table, or a keyword or alpha literal.

Operation - Resources are initialized at beginning of simulation to the values given above.

Relationships - Resources are used by Processes with the ALLOC, DEALLOC, RESET, LOCK, UNLOCK and TEST Primitives.

Resource entities are entered using the Design User Interface EDIT command (see section 6.1.4).

3.7 ACTION

The Action entity represents time consumption for any activity, decision, etc., that consumes time. This entity functions in conjunction with the ACTION Primitive. For each defined Action entity, statistics on the time consumed by the associated ACTION Primitive are collected for the simulation's Action report. For this reason, each Action named in an ACTION Primitive is given a separate definition outside the Process in which it appears.

In the form for this definition, the ACTION field contains a name identical with one that appears in a Process. The field CLASS is optional and is intended as a means to document what kind of activity is taking place or who/what is performing the action (viz., man/machine). DESCRIPTION is used for any mnemonic. The form for the Action entity is shown in figure 3-7.

Figure 3-7. Form for the Action Entity

Following is a description of the fields in the Action form:

ACTION: 1 to 8 character name of action

CLASS: user defined class

DESCRIPTION: Any user comment (0 to 53 characters)

Relationships - Actions are referenced by the ACTION Primitive.

Action entities are entered using the Design User Interface EDIT command (see section 6.1.4).

3.8 PROCESS

The Process entity is used to represent the sequential logic and activities, operations, functions, etc., of the modeled system. Processes are composed of Primitives, each of which represents a step in the function being modeled by the Process. It is at the Primitive level that Resources are allocated and deallocated, time is consumed, decisions take place, etc.

In the graphic representation of a Process, the Primitives are flanked at the top and bottom by figures labeled START and END. These figures represent the logical entry and exit points for the Process.

Processes are initiated by (1) Scenarios and Loads (within Scenarios) and (2) by other Processes through the CALL and SEND Primitives. Once initiated, the execution of the Process depends upon the availability of the Resources that the Process references through the ALLOC and DEALLOC Primitives.

There are three types of Processes: parameter passing, Item passing, and standard. Each differs in how it is triggered.

A parameter passing Process is one that takes values of local variables from another Process as inputs and/or returns the values of local variables to the other Process as outputs. Such Processes can be triggered only by a CALL Primitive and it is the calling Process which sets up the relation for the values given and returned (see CALL Primitive, section 3.9.5). The given and return values can be numerics, string literals, keywords, the names of Items, Queues, Resources, Processes, Tables and Actions.

An Item passing Process is one that is triggered by having Item(s) delivered to it from other Process(es) through the SEND Primitive. The required Items need not be delivered from a single Process; the sending Processes may be as many as six, but the Process will not execute until all of the Items indicated in the definition are delivered.

A standard Process is one which neither requires Items nor is given (or returns) parameters. It may be triggered by either a CALL Primitive from another Process or through the Scenario or Loads.

When a Process is defined, the node in which the Process is to execute is specified. If the Process can execute in any node, or if there is no architecture, ALL can be specified. Generally, when a Process is triggered, it executes in the same node as its parent, or when a Process is triggered from a Load, the Load nodes specify where the Process is to execute. However, if a Process is triggered from a Scenario, the node specified for the Process is the one in which the Process executes. The node specified in the Process definition is also available to the user through the $NODE keyword (see section 3.16).

The initial form for the Process entity is shown in figure 3-8.

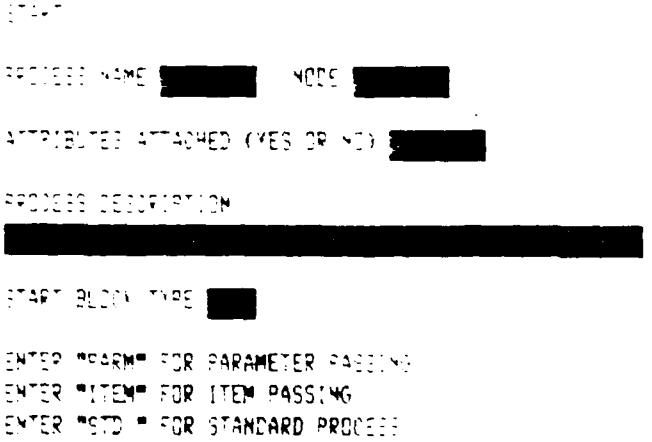


Figure 3-8. Initial Form for the Process Entity

Following is a description of the fields in the initial Process entity form:

PROCESS NAME:	1 to 8 character name of Process
NODE:	architecture node in which this Process is to execute (if its execution is restricted to a specific node; ALL in this field indicates the Process may execute in any node)
ATTRIBUTES ATTACHED:	YES or NO to indicate whether the Process has attributes.
PROCESS DESCRIPTION:	0 to 53 alphanumeric character description.
START BLOCK TYPE:	(STD, ITEM, PARM)

To define an Item passing Process the user enters "Item" in the START BLOCK TYPE field. The user will then be presented with the form shown in figure 3-9.

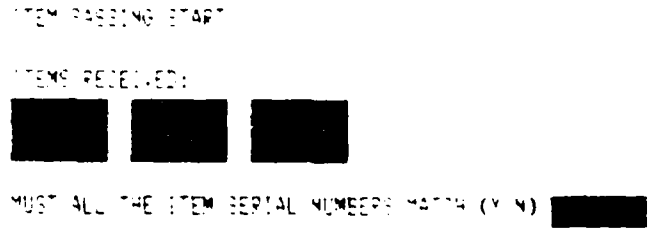


Figure 3-9. Form for an Item Passing Process

This form is for providing a list of the needed Items. The Items received by each must be of distinct types.

The field concerning the matching of serial numbers asks whether the TAIL numbers (which is a default attribute of every Item) must be the same for all the Items in the Process. If the user enters "Yes" in this field, the Process will not execute until it has received Items of the specified type to which the same TAIL number attribute has been assigned.

To define a parameter passing Process the user enters "PARM" in the START BLOCK TYPE field. The user will then be presented with the form shown in figure 3-10.

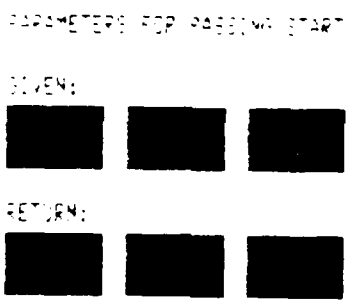


Figure 3-10. Form for Parameter Passing Process

This form is for providing the names of the local variables to be given and returned to any Process that calls it through the CALL Primitive. The CALL Primitive must contain the same number of entries in its given and return lists as the called Process. If the CALL Primitive does not give or return all the necessary values, an execution error will occur indicating a disagreement in the number of values.

To define a standard Process the user enters "STD" in the START BLOCK TYPE field. Since no inputs are relevant to its execution, there is no secondary form for the definition of a standard Process.

Figure 3-11 is a typical flowchart representation of a Process. This graphical representation of the logic of a Process is presented to the user during the design of an AISIM model.

Relationships - Processes are constructed from Primitives. Resources are used by the Process through the ALLOC, DEALLOC, RESET, LOCK, UNLOCK, and TEST Primitives. Time is consumed by the ACTION Primitive. Processes are initiated by Loads, Scenarios and by other Processes through the CALL and SEND Primitives.

Process entities are entered using the Design User Interface EDIT command (see section 6.1.4).

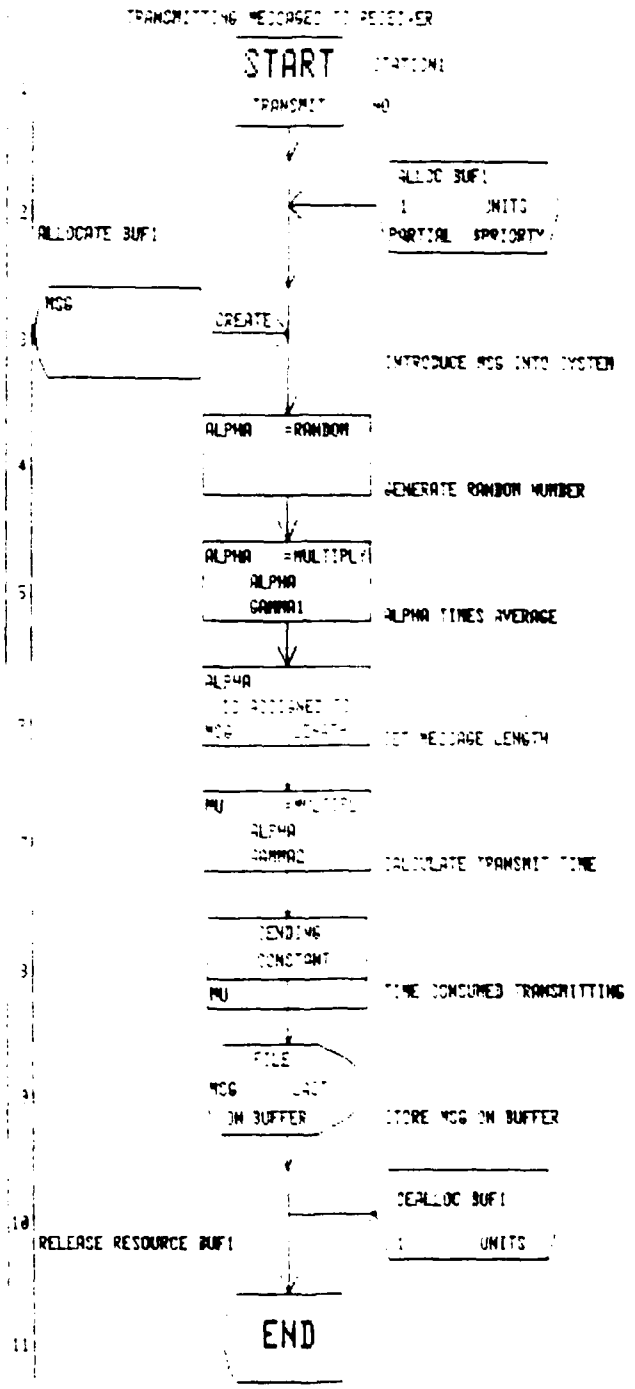


Figure 3-11. Sample Process Diagram

3.9 PRIMITIVES

Primitives are the constituent elements of Processes and are used to characterize procedural steps by sequential logic. AISIM offers a list of 25 Primitives. Although limited in number, the Primitives have been shown to represent all logical operations for computer system modeling. The Primitives can be grouped into nine functional categories. These categories are as follows:

Process Execution Control

CALL
SEND
SUSPEND
RESUME
WAIT

These Primitives control the initiation and sequencing of Processes.

Branch Control

COMPARE
BRANCH
ENTRY
PROB
LOOP

These Primitives govern the internal branching in the logic of a Process.

Item Handling

CREATE
DESTROY

These two Primitives govern the introduction and elimination of a system's transient data elements.

Time Consumption

ACTION

This Primitive represents the consumption of time through some activity, decision, etc.

Mathematical Operations

EVAL

This Primitive governs calculations, invoking standard mathematical functions and operations or making use of user-defined Tables.

Resource Allocation

ALLOC
DEALLOC
RESET
TEST
LOCK
UNLOCK

These Primitives govern the use of Resources.

Queue Manipulation

FILE
FIND
REMOVE

These Primitives govern storage and retrieval on Queues.

Variable Assignment

ASSIGN

This Primitive governs the assignment of values to Variables or Attributes (both numerical and non-numerical).

Debugging

TRACE

This Primitive has the special function of creating a record of the sequence of Process Primitive executions which takes place during simulation. It is used for debugging and validating a model.

Figure 3-12 shows the graphic representation of each Primitive, and following is a description of the meaning of each Primitive and the parameters necessary to define each. Primitives are entered using the Process Editor Interface of the Design User Interface (see section 6.2).

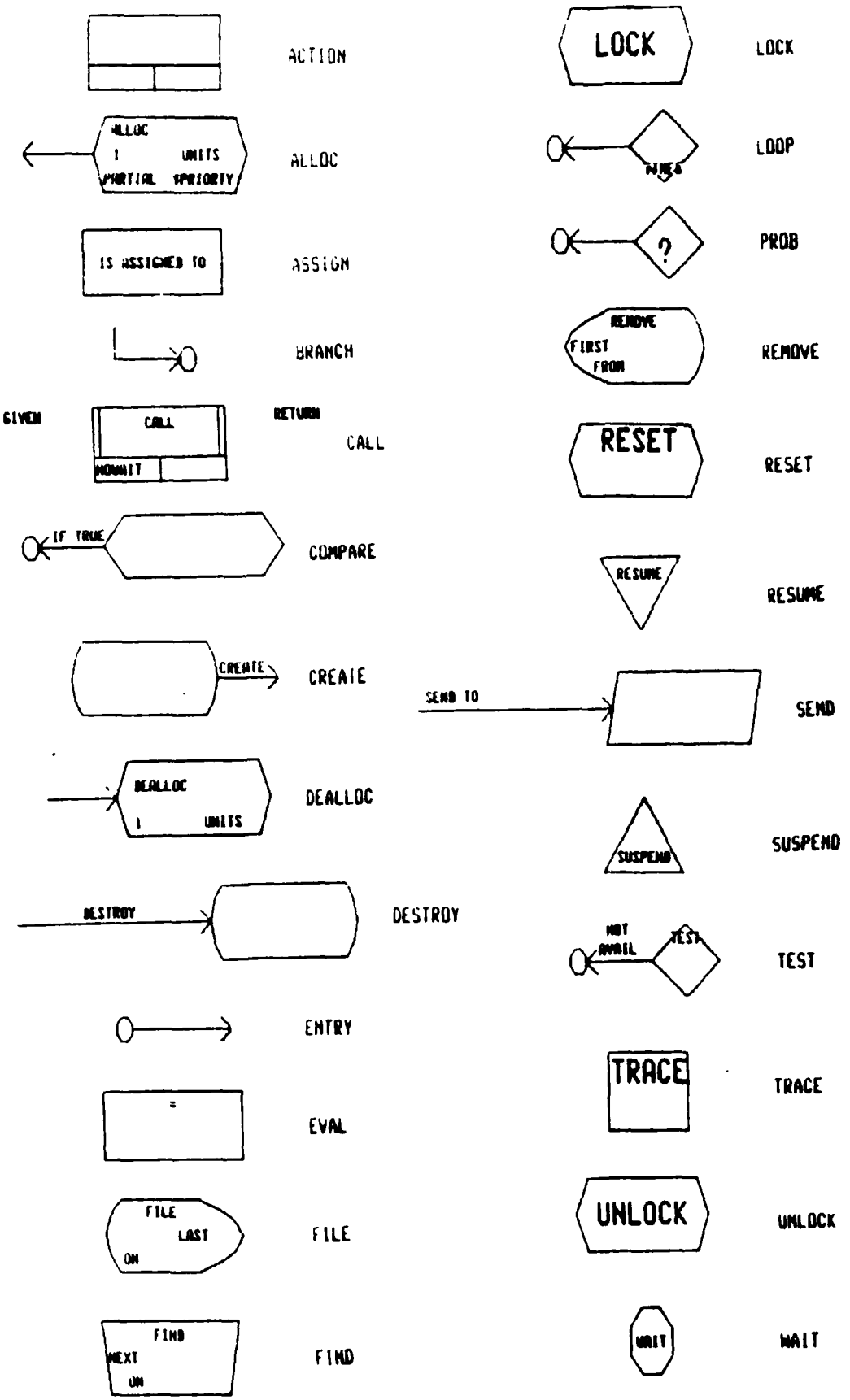


Figure 3-12. Graphical Representations of Process Primitives

3.9.1 ACTION

The ACTION Primitive represents the consumption of time for an activity that consumes time. The ACTION Primitive is used to model the time to perform some real work event such as a man's activity or a machine's activity. The time consumed by an ACTION Primitive is determined according to the selected distribution function (described below). The form for an ACTION Primitive is shown in figure 3-13.

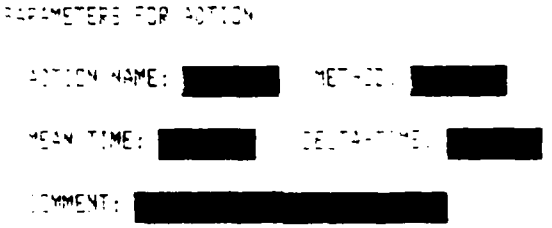


Figure 3-13. Form for an ACTION Primitive

Following is a description of the fields of an ACTION form:

ACTION NAME: A reference to a defined Action entity

METHOD: Distribution function type, which may be: CONSTANT, EXPONENT, LOGNORML, NORMAL, UNIFORM, GAMMA, ERLANG or WEIBULL. (The random number seed used for statistical functions can be controlled by the user in the AUI.)

MEAN TIME: Typically specifies the average duration time of the Action. This parameter varies in meaning depending on the METHOD selected. For CONSTANT, it specifies the exact duration value. For WEIBULL, it specifies the distribution's scale parameter. For all other methods, it specifies the mean duration.

DELTA TIME: This parameter varies in meaning depending on the METHOD selected. Typically it specifies the variation, about the mean, in the duration times. Specifically:

CONSTANT - inapplicable (i.e., leave field blank)

EXPONENT - inapplicable (i.e., leave field blank)

LOGNORML - standard deviation

NORMAL - standard deviation

UNIFORM - range about the mean (i.e., the difference between the largest possible duration and the mean duration).

GAMMA - K

ERLANG - order of distribution function

WEIBULL - shape parameter

COMMENT: Any user comment.

3.9.2 ALLOC

The ALLOC Primitive indicates the allocation of (request to use) a Resource which is needed by the Process. Whether a Resource requested by the ALLOC Primitive is actually obtained by a Process depends on a number of conditions, as described in the section on the Resource entity, section 3.6. If a Resource unit is in the idle state, it is available to be allocated to the requesting Process. If the Resource is busy, then allocated Resource units are checked to see if a Process can be preempted by priority (priority is inverse - priority 1 preempts priority 2) unless the Resource is protected with a LOCK primitive. The form for the ALLOC Primitive is shown in figure 3-14.

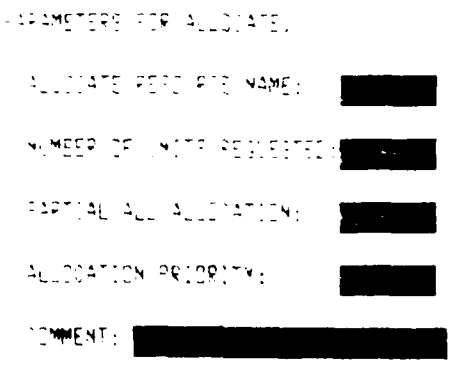


Figure 3-14. Form for the ALLOC Primitive

Following is a description of the fields in the ALLOC form:

ALLOCATE RESOURCE NAME: A reference to a Resource

NUMBER OF UNITS REQUESTED: The number of Resource units to be allocated.

PARTIAL/ALL ALLOCATION: This specifies whether the Resource units will be allocated as they become available (PARTIAL) or only allocated simultaneously when they are all available (ALL).

ALLOCATION PRIORITY: The priority to be used to determine which allocation request will be satisfied in the case of Resource contention. SPRIORTY is the default and evaluates to the priority of this Process.

COMMENT: Any user comment.

3.9.3 ASSIGN

The ASSIGN Primitive is used to set the value of the following references:

1. a global Variable
2. a local (to the executing Process) variable
3. the attribute of an Item (currently attached to the Process)
4. the attribute of a Resource
5. $CNODE (see section 3.16)
6. the attribute of a Process

Values that can be accessed for the assignment are:

1. signed, single precision, real or integer numbers
2. $CLOCK (see section 3.16)
3. global Variables or Constants
4. local variables
5. Resources with any of the qualifiers NWAITQ, NBUSYQ, NINACTQ or NIDLEQ (see section 3.12)
6. Item attribute values
7. Queue qualifiers NQUEUE or TQUEUE (see section 3.12)
8. Resource attribute values
9. Process attribute values
10. an Item name
11. a Resource name
12. a Process name
13. a Queue name
14. a Table name
15. an Action name
16. $NODE (see section 3.16)

17. $NXTNODE (see section 3.16)

18. $LINK (see section 3.16)

19. $TASK (see section 3.16)

20. $CNODE (see section 3.16)

21. an alpha literal (first character is $) (see section 3.15)

The form for the ASSIGN Primitive is shown in figure 3-15.

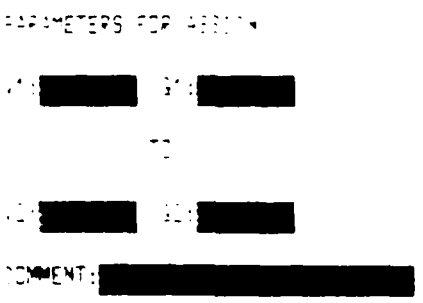


Figure 3-15. Form for the ASSIGN Primitive

In the form, V1 and Q1 are used to reference the current value, and V2 and Q2 are used to reference the value being set. For accessing values such as local variables, the simulation clock, etc., only the "V" fields need to be used. If the user is accessing an attribute of an entity, such as an Item, both the "V" and "Q" fields need to be used. The "V" field contains the name of the entity (Item, etc.) being accessed, and the "Q" field contains the name of the attribute of the entity whose value is desired or being set.

Following are examples of some typical entries:

V1: Item	V1: Item	V1: Variable
Q1: attribute	Q1: attribute	Q1:
V2: Item	V2: Variable	V2: Item
Q2: attribute	Q2:	Q2: attribute
V1: Variable	V1: Constant	V1: Constant
Q1:	Q1:	Q1:
V2: Variable	V2: Item	V2: Variable
Q2:	Q2: attribute	Q2:

COMMENT: Any user comment.

Note that it is the entity specified by V2 and Q2 that takes on the new value specified by V1 and Q1.

3.9.4 BRANCH

The BRANCH Primitive indicates an unconditional branch to a named entry point. It is used for Process execution sequence control. The form for the BRANCH Primitive is shown in figure 3-16.

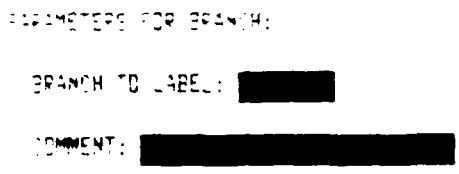


Figure 3-16. Form for the BRANCH Primitive

Folowing is a description of the fields in the BRANCH form:

LABEL: The entry point to which the Process execution is to go (which must be defined by an ENTRY Primitive).

COMMENT: Any user comment.

3.9.5 CALL

The CALL Primitive triggers execution of the called Process.

A CALL has one of three options (1) WAIT, (2) NOWAIT and (3) BLOCK. If a Process is called with the option WAIT, the calling Process will suspend execution until the called Process is completed. If a Process is called with the NOWAIT option, both called and calling Processes will execute simultaneously and will have no further communication. If a Process is called with the BLOCK option, the two Processes will execute in parallel until a WAIT Primitive is reached in the execution of the calling Process. When the WAIT Primitive is reached, the calling Process suspends execution until the called Process(es) complete(s). The principal purpose of the BLOCK option is to allow the calling of several different Processes, all of which must be completed before the calling Process will continue. If several Processes are called with the BLOCK parameter, the calling Process will suspend at the WAIT Primitive--whose presence somewhere below such a CALL Primitive is obligatory--until all of them have completed execution.

Two of the three kinds of Processes can be triggered via the CALL Primitive: parameter passing Processes and standard Processes. The triggering of an Item passing process is discussed in the section describing the SEND primitive. In triggering a parameter passing Process with a CALL Primitive, parameters are given to the called Process and/or parameters are returned to the calling Process. Parameters can be numerics, string literals, keywords, or the names of Items, Queues, Resources, Processes, Tables, and Actions. Parameter passing Processes with return parameters can be called only with the WAIT option. Standard Processes, which neither give nor return information may be called with any of the three options WAIT, NOWAIT and BLOCK.

The CALL also requires that a priority be established for the called Process. Priority is inverse, priority 1 preempts priority 2. This priority may be used by the called Process when competing with other Processes for available Resources (through the ALLOC Primitive with $PRIORTY, see section 3.9.2).

The form for the CALL Primitive is shown in figure 3-17.

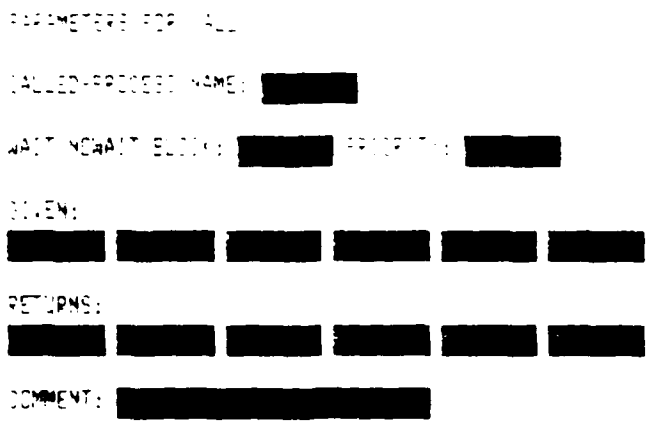


Figure 3-17. Form for the CALL Primitive

Following is a description of the fields in the CALL form:

CALLED-PROCESS NAME:	The Process to be triggered.
WAIT/NOWAIT/BLOCK:	Explained above.
PRIORITY:	The priority associated with the triggered Process (discussed above).
GIVEN:	Up to six parameters whose values are to be communicated to the called Process. Left blank if called Process is a standard Process.
RETURN:	Up to six parameters whose values are to be returned to the calling Process. Left blank if called Process is a standard Process.
COMMENT:	Any user comment

3.9.6 COMPARE

The COMPARE Primitive is used to model decisions based on user-controlled variables or the values of system keywords and attributes. The COMPARE performs the following operation:

IF P IS TRUE, THEN GO TO A

where:

"A" is an ENTRY label (defined by an ENTRY primitive) which is branched to if P is true.

"P" is a predicate which can be TRUE or FALSE. It consists of a phrase:

X1 OP X2

X1,X2 can be:

(1) signed, single precision, real or integer numbers

(2) global Variables or Constants

(3) local Variables

(4) Resources with either NWAITQ, NBUSYQ, NINACTQ or NIDLEQ attributes (which cannot be modified by the user) (see section 3.12)

(5) SCLOCK (see section 3.16)

(6) a value specified by an Item name and attribute

(7) a value specified by a Resource name and attribute

(8) a value specified by a Process name and attribute

(9) an Item name

(10) a Resource name

(11) a Process name

(12) a Queue name

(13) a Table name

(14) an Action name

(15) SNODE (see section 3.16)

(16) SNXTNODE (see section 3.16)

(17) $LINK (see section 3.16)

(18) $TASK (see section 3.16)

(19) $CNODE (see section 3.16)

(20) an alpha literal (first character is $) (see section 3.15)

(21) a Queue with either NQUEUE or TQUEUE as an attribute (which cannot be modified by the user) (see section 3.12)

"OP" is one of the following operators:

EQ - equal to,

NE - not equal to,

GE - greater than or equal to,

GT - greater than,

LE - less than or equal to,

LT - less than.

Operation - "X1" is compared to "X2" using real, single precision arithmetic. If the comparison results in the same relation as "OP", then "P" is set TRUE and a branch is made to label "A"; otherwise, no branch is made (the next Process Primitive is executed).

The form for the COMPARE Primitive is shown in figure 3-18.

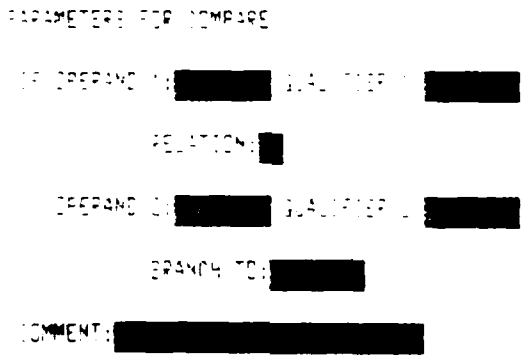


Figure 3-18. Form for the COMPARE Primitive

The parameters of the form are filled in as indicated above.

3.9.7 CREATE

The CREATE Primitive is used to create Items (note the SEND Primitive can also create Items as part of its function). The initial attribute values (defined when the Item is declared) are assigned upon creation. Each Item created is attached to the Process. Two Items of the same name cannot exist in a Process at the same time. Item definitions are specified in the DUI. The form for the CREATE Primitive is shown in figure 3-19.

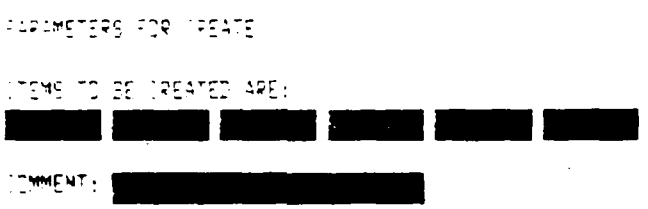


Figure 3-19. Form for the CREATE Primitive

Following is a description of the fields in the CREATE form:

ITEMS: references to distinct Item types, instances of which are to be created.

COMMENT: Any user comment.

3.9.8 DEALLOC

The DEALLOC Primitive indicates the release of previously allocated Resources. It is used to represent the release of a Resource (making it available to another request) upon completion of a job. The form for the DEALLOC Primitive is shown in figure 3-20.

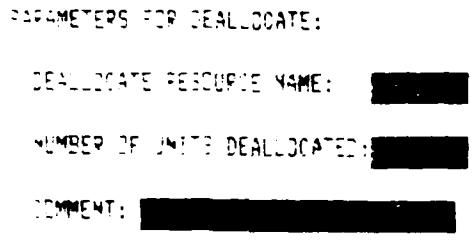


Figure 3-20. Form for the DEALLOC Primitive

Following is a description of the fields in the DEALLOC form:

RESOURCE NAME: A reference to the Resource to be released.

NUMBER OF UNITS: A reference to the integer number of Resource units to be returned to the idle state.

COMMENT: Any user comment.

3.9.9 DESTROY

The DESTROY Primitive is used to eliminate Items from the system, marking the end of the time in system. When an Item is destroyed, statistics on its time in the system are tabulated for the simulation's Item report.

The form for the DESTROY Primitive is shown in figure 3-21.

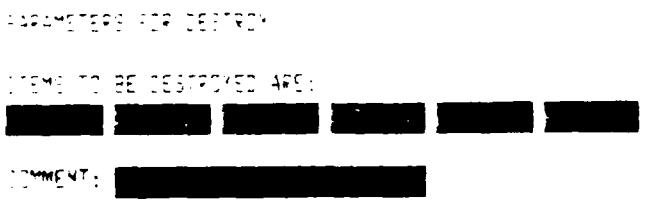


Figure 3-21. Form for the DESTROY Primitive

Following is a description of the fields in the DESTROY form:

ITEMS: References to distinct Item types, instances of which are to be destroyed.

COMMENT: Any user comment.

3.9.10 ENTRY

The ENTRY Primitive is used to define entry points from the branching Primitives, BRANCH, PROB, COMPARE, TEST and LOOP. The form for the ENTRY Primitive is shown in figure 3-22.

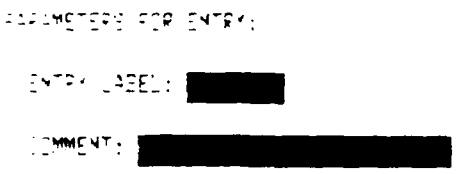


Figure 3-22. Form for the ENTRY Primitive

Following is a description of the fields in the ENTRY Primitive:

ENTRY LABEL: The 1-8 character name of the entry point used by the branching Primitive(s) which transfer control to it.

COMMENT: Any user comment.

3.9.11 EVAL

The EVAL Primitive is used to perform simple arithmetic functions within a Process so that model logic and timing can be a function of variables rather than a constant. EVAL operates in the following manner:
X = f(a,b)

where:

X is any variable that is changed to the value f(a,b)

"a" and "b" are arguments that can be

1. signed, single precision, real, or integer number, or
2. named Variable or Constant, or
3. named local variable, or
4. $CLOCK (simulation clock value)

where f is one of 27 functions below. All calculations are carried out in a single precision, real arithmetic.

	FUNCTION NAME	RESULT
1.	ADD	a+b
2.	SUBTRACT	a-b
3.	MULTIPLY	a*b
4.	DIVIDE	a/b
5.	ABSOLUTE	|a|
6.	INTEGER	returns the integer part of a number
7.	POWER	a**b
8.	COSINE	cos(a) (for a in radians)
9.	SINE	sin(a) (for a in radians)
10.	TANGENT	sin(a)/cos(a)
11.	SQRT	sqrt a
12.	RANDOM	random fraction (random number between 0 and 1.0)
13.	ARCOSINE	arcosine(a) (in radians)
14.	ARCSINE	arcsine(a) (in radians)
15.	ARCTAN	arctangent(a/b) (in radians)
16.	BETA	random sample of the beta function with a = power of x; a>0 b = power of 1-x; b>0
17.	BINOMIAL	random sample of the binomial function with a = number of trials b = probability of success
18.	ERLANG	random sample of Erlang function with a = mean b = k (integer order of function)

19.	EXPONENT	random sample of exponential function with a = mean
20.	GAMMA	random sample of gamma function with a = mean b = k
21.	LOGE	natural logarithm of a; a>0
22.	LOGNORML	random sample of log normal function with a = mean b = standard deviation
23.	LOG10	common logarithm of a; a>
24.	NORMAL	random sample of normal function with a = mean b = standard deviation
25.	POISSON	random sample of Poisson function with a = mean
26.	UNIFORM	random sample of a uniform function with a = mean b = delta (i.e., the difference between the largest possible value and the mean value)
27.	WEIBULL	random sample of the Weibull function with a = scale parameter b = shape parameter

In addition to these functions the user may define his own functions through the Table entity (see the section on Tables). The form for the EVAL Primitive is shown in figure 3-23.

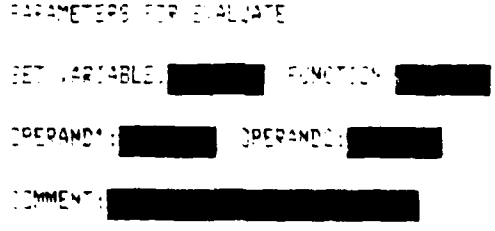


Figure 3-23. Form for the EVAL Primitive

Following is a description of the fields in the EVAL form:

VARIABLE: The local variable whose value is to be set.

FUNCTION: The operation used to calculate the value of the variable.

OPERAND1: The first operand in the calculation of the new variable ("a" parameter). This may be blank, depending on the function.

OPERAND2: The second operand in the calculation of the new variable ("b" parameter). This may be blank, depending on the function.

COMMENT: Any user comment.

3.9.12 FILE

The FILE Primitive is used to place an Item on a user-defined Queue.

The effect of filing an Item on a user-defined Queue is to keep it in storage after the Process from which it is filed has ceased execution. The form for the FILE Primitive is shown in figure 3-24.

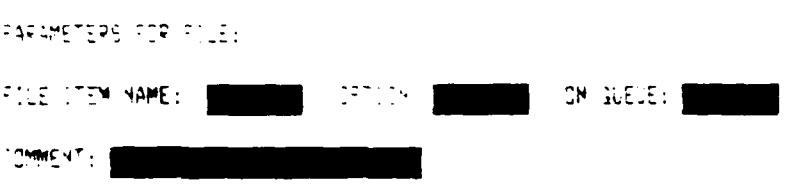


Figure 3-24. Form for the FILE Primitive

Following is a description of the fields in the FILE form:

FILE ITEM NAME: A reference to the Item to be filed.

OPTION: The location in the Queue at which the entity is to be filed relative to the Queue pointer. The following can be used:

FIRST - The entity is placed first and the Queue pointer is set to it.

LAST - The entity is placed last and the Queue pointer is set to it.

NEXT - The entity is placed after the current Queue pointer position in the Queue and the Queue pointer is reset to it.

BEFORE - The entity is placed before the current Queue pointer position in the Queue and the Queue pointer is reset to it.

QUEUE: The Queue on which the Item is to be filed.

COMMENT: Any user comment.

3.9.13 FIND

The FIND Primitive is used to reset the Queue pointer on a user-defined Queue (section 3.4) or a cross-reference set (section 3.5.2), and to assign to a local variable a "locator" pointer to a current position in the Queue. The rules governing Queue pointers are covered above in the section on user-defined Queues. The form for the FIND Primitive is shown in figure 3-25.

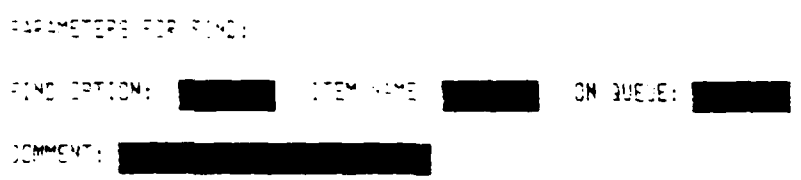


Figure 3-25. Form for the FIND Primitive

Following is a description of the fields in the FIND form:

FIND OPTION: The location (FIRST, LAST, NEXT, or BEFORE) of the Item or member of the cross-reference set to be assigned to the variable relative to the present Queue pointer.

ITEM NAME: The local variable which will refer to the Item or member of a cross-reference set.

ON QUEUE: The name of the Queue or cross-reference set that is to be traversed. If the cross-reference set is intended, the entity type whose cross-reference set is to be traversed is entered.

COMMENT: Any user comment.

The effect of locating an element with the FIND Primitive is (1) to set the Queue pointer to the beginning or end of the ordered holding area (i.e., FIRST or LAST) or relative to the previous location of the Queue pointer (i.e., NEXT or BEFORE), and (2) to assign the element in the position then indicated to the local variable.

3.9.14 LOCK

The LOCK Primitive prevents a Process from being suspended by losing Resources to a "higher" priority Process (priority is inverse, priority 1 preempts priority 2). LOCK is used to represent uninterruptable work. If LOCK is not used, Process execution can be suspended by a higher priority Process. When a Process loses any one of the Resources it has allocated it stops execution and is placed on a system-defined queue (the wait queue) until the Resource is again available. The LOCK Primitive overrides this suspension. The form for the LOCK Primitive is shown in figure 3-26.

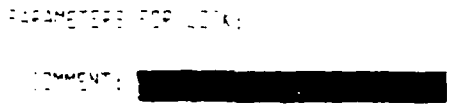


Figure 3-26. Form for the LOCK Primitive

Following is a description of the field in the LOCK form:

COMMENT: Any user comment.

3.9.15 LOOP

The LOOP Primitive causes a branch to a named entry point a specified number of times. The form for the LOOP Primitive is shown in figure 3-27.

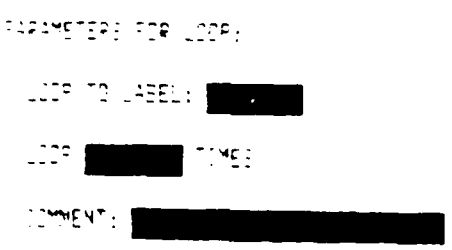


Figure 3-27. Form for the LOOP Primitive

Following is a description of the fields in the LOOP form:

LABEL: The name of the ENTRY label (defined by an ENTRY Primitive) to which execution is to branch.

LOOP: Indicates the number of times Primitives between the ENTRY label and the LOOP Primitive will be executed. This includes the initial pass. For example, if 10 was used, then for each execution of the Process, the Primitives between the Entry label and the LOOP Primitive would be executed 10 times. Execution control would branch back to the ENTRY label 9 times.

COMMENT: Any user comment.

3.9.16 PROB

The PROB Primitive is used to model stochastic decision making. It indicates a probabilistic branch to a named entry point. Random number selection for the probabilistic branch can be controlled by the use of the EDIT STREAM command in the AUI. The form for the PROB Primitive is shown in figure 3-28.

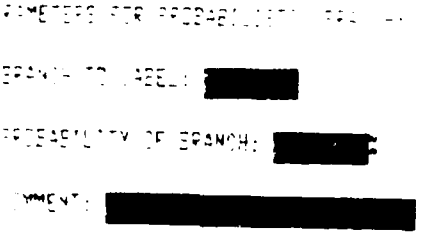


Figure 3-28. Form for the PROB Primitive

Following is a description of the fields in the PROB form:

LABEL: The ENTRY label (defined by an ENTRY Primitive) to which the branching is to take place.

PROBABILITY: The probability with which the branching is to take place, expressed in (integer) percent.

COMMENT: Any user comment.

3.9.17 REMOVE

The REMOVE Primitive is used to remove an Item from a user-defined Queue.

The effect of removing an Item is to make it inaccessible to other Processes until it has been placed on another Queue (through the FILE Primitive) or delivered to another Process through the SEND Primitive. The form for the REMOVE Primitive is shown in figure 3-29.

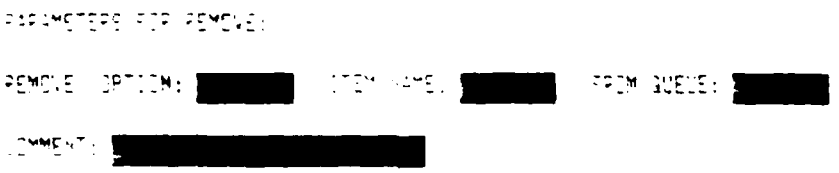


Figure 3-29. Form for the REMOVE Primitive

Following is a description of the fields in the REMOVE form:

REMOVE OPTION: The location in the Queue of the Item to be removed. The option can be one of the following:

- FIRST - The first entity is removed and the Queue pointer is reset to the new first element.
- LAST - The last entity is removed and the Queue pointer is reset to the new last element.
- NEXT - The entity associated with the current Queue pointer location is removed and the Queue pointer is reset to the succeeding element to it in the Queue.

ITEM NAME: The local variable that will contain the Item which is removed from the Queue. If there is no Item to be removed, this local variable is set to zero.

FROM QUEUE: The Queue from which the Item is to be removed.

COMMENT: Any user comment.

3.9.18 RESET

The RESET Primitive redefines the number of available units of a named Resource to plus or minus the indicated value. It is used to represent the increase or decrease of the available number of Resource units. The form for the RESET Primitive is shown in figure 3-30.

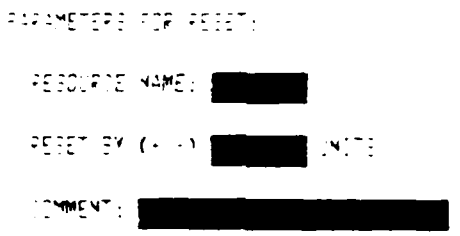


Figure 3-30. Form for the RESET Primitive

Following is a description of the fields in the RESET form:

RESOURCE: A reference to a Resource whose available units are increasing or decreasing.

RESET BY (+/-): The number of units to be added to or subtracted from those presently available. If more units are to be made available, this value is positive. If units are to be made unavailable, this value is negative.

COMMENT: Any user comment.

3.9.19 RESUME

The RESUME Primitive is used to control explicitly the resumption of a Process which has been suspended through the SUSPEND Primitive. Resources deallocated at the time of suspension must be obtained again before Process execution progresses. The requests for these Resources is automatically handled by the RESUME Primitive. There is no preferential treatment given to these requests. They are treated in the same manner as an ALLOC Primitive. The form for the RESUME Primitive is shown in figure 3-31.

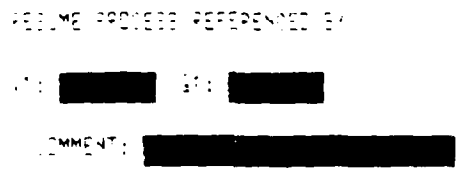


Figure 3-31. Form for the RESUME Primitive

The fields V1 and Q1 constitute a reference to task that is being resumed (see SUSPEND) and the COMMENT field is any user comment.

3.9.20 SEND

The SEND Primitive is used to send up to six Items to an Item passing Process. If an Item to be sent is not currently attached to the sending Process, it is automatically created. When the Items are sent, the receiving Process determines whether all the Items required by its definition have been received. If they have, the Process then initiates; if not, it will wait until all of the necessary Items have been received before executing. The form for the SEND Primitive is shown in figure 3-32.

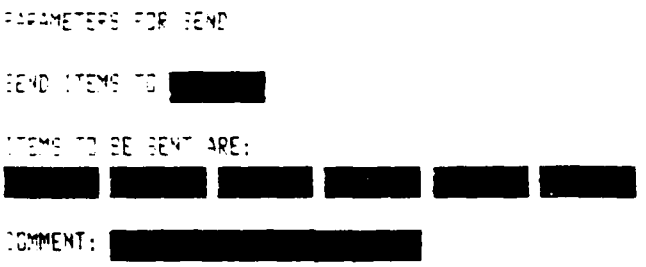


Figure 3-32. Form for the SEND Primitive

Following is a description of the fields in the SEND form.

SEND: A reference to the Process to which Items are to be sent.

ITEMS: References to up to six Item types, instances of which are to be sent.

COMMENT: Any user comment.

3.9.21 SUSPEND

The SUSPEND Primitive is used to suspend the Process in which it appears. A Process that suspends itself with this Primitive may only be resumed by another Process which uses the RESUME Primitive. Since the RESUME Primitive must be able to refer to the task instant to be resumed, the suspending Process instance must save a reference to itself (i.e., assign the value of the keyword $TASK to a global Variable or send it as an attribute of an Item) for later access by a RESUME Primitive. See section 3.16 for a description of $TASK. The SUSPEND Primitive causes the deallocation of the Resources allocated to the Process. The form for the SUSPEND Primitive is shown in figure 3-33.

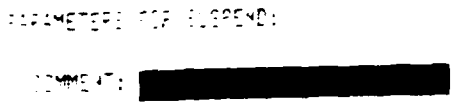


Figure 3-33. Form for the SUSPEND Primitive

Following is a description of the field in the SUSPEND form:

COMMENT: Any user comment.

3.9.22 TEST

The TEST Primitive indicates a branch to a named ENTRY Primitive if a Resource or Queue is not available. It is used to model decision making based on the availability of needed Resources or Queues. The form for the TEST Primitive is shown in figure 3-34.

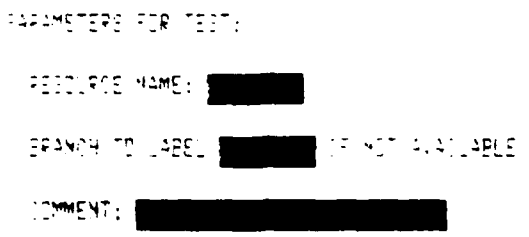


Figure 3-34. Form for the TEST Primitive

Following is a description of the fields in the TEST form:

RESOURCE NAME: A reference to the Resource or Queue being tested for availability.

BRANCH TO LABEL: The name of the ENTRY label (defined by an ENTRY Primitive) to which execution is to branch if the Queue or Resource is not available.

COMMENT: Any user comment.

3.9.23 TRACE

The TRACE Primitive starts a debugging mechanism that is useful for analyzing the dynamics of an AISIM model. The effect of the TRACE Primitive is to create a file that records every execution of a Process and of the following Primitives within the Process.

START

CALL

ALLOC

DEALLOC

END

RESUME

RESET

SUSPEND

TRACE (on or off)

These Primitives are traced because they introduce major changes in the state of the system into a simulation run.

When the TRACE Primitive is operating, every instance of these Primitives in every Process is recorded either for the remainder of the simulation or until TRACE is turned off. The trace line writes out the simulation clock time, the node in which the Primitive is executed, and the Process executing the Primitive. The format for a trace line is the following:

T = clock time N = node name P = Process name Primitive parameter

The form for the TRACE Primitive is shown in figure 3-35.

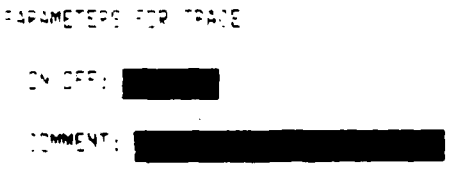


Figure 3-35. Form for the TRACE Primitive

Following is a description of the fields on the TRACE form:

ON/OFF: "ON" to enable the TRACE.

"OFF" to disable the TRACE.

COMMENT: Any user comment.

3.9.24 UNLOCK

The UNLOCK Primitive cancels the effect of a previously executed LOCK Primitive. It is used to represent the conclusion of the uninterruptable phase of a Process. The form for the UNLOCK Primitive is shown in figure 3-36.

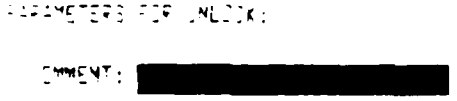


Figure 3-36. Form for the UNLOCK Primitive

Following is a description of the field in the UNLOCK Primitive:

COMMENT: Any user comment.

71

3.9.25 WAIT

The WAIT Primitive is used in conjunction with the CALL Primitive when the BLOCK option is used. The WAIT Primitive indicates that the calling Process is to be suspended until all Processes it triggered by a CALL with the BLOCK option have completed and returned control to the calling Process. It is generally used to model phenomena such as assembly points, executive schedulers, and other events in which progress cannot continue until several parallel activities are completed. Resources currently in possession of the calling Process are not deallocated. The form for the WAIT Primitive is shown in figure 3-37.

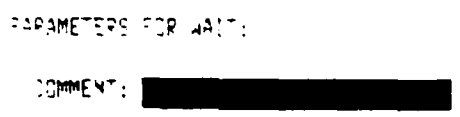


Figure 3-37. Form for the WAIT Primitive

Following is a description of the field in the WAIT Primitive:

COMMENT: Any user comment.

3.10 LEGAL PATH TABLE - NODE - LINK

The Legal Path Table (LPT) entity is the means by which the user can model physical communication paths between Resources. Typically, this is referred to as inter-node communication. When the LPT is not used, the communication mechanisms are implicit in the Process logic and do not usually have explicit Resources that cause communication queueing and transfer delays.

Two other model elements need to be discussed as part of the LPT entity; these are nodes and links. Nodes represent the points in an architecture where processing occurs. Links are the communication paths between nodes. Each node and link is actually a model Resource -- the name of the Resource being the name of the node or link. Full duplex links (denoted by ".F" after the link name) are two Resources. One will be named the name of the link with ".A" appended to it and the other with ".B".

The LPT consists of a four part list that specifies the FROM node, a TO node, a NEXT node, and a LINK. An example of Legal Path Table entries is given in figure 3-38.

FROM NODE	TO NODE	NEXT NODE	VIA LINK
A	C	C	C1
B	C	C	C2
C	A	A	C1
C	B	B	C2
C	D	D	C3
D	C	C	C4
D	E	E	C6
D	F	F	C5
E	D	D	C8
E	G	G	C8
F	D	D	C5
F	G	G	C7
G	E	E	C8
G	F	F	C7
G	H	H	C9
G	I	I	C10
H	G	G	C9
I	G	G	C10

Figure 3-38. Sample Legal Path Table Entries

The headings indicate that to move from the FROM node to the TO node one must first go to the NEXT node via the LINK.

The LPT is a passive entity in that it does not contribute directly to the simulation statistics but, instead, is simply a table of values used by a model to effect data flow through a system. It is only changed through the Architecture Design Editor and therefore remains constant for any specific simulation run. Processes reference the LPT through the ASSIGN or COMPARE Primitives using $CNODE (current node), $NXTNODE (next node as

specified in LPT), and $LINK (the Link for the transfer) keywords (see section 3.16).

Operation - Every Process in AISIM can be set to execute in a specific node. Using the LPT through the keywords and the ASSIGN Primitive, a Process can locate itself in the network and reference other nodes. The referencing is done symbolically so that a Process can do this when executing. This allows AISIM to model different architectures without changing the model Processes.

3.11 TABLES

Tables are user definable functions with 1 to 15 entries. Each entry consists of an X-VALUE and a Y-VALUE. The following may be used for these parameter values: (1) both numeric, (2) one numeric and the other alphanumeric or (3) both alphanumeric.

Tables are accessed by using the EVAL Primitive. The EVAL FUNCTION parameter is the name of the desired Table. Operand 1 is the X-VALUE. Operand 2 is not used. The SET VARIABLE will be set to the Y-VALUE which maps from the X-VALUE.

3.11.1 Discrete Tables

If the Table accessed is discrete (TYPE is D), the Table entry's X-VALUE must be numeric, and the X-VALUE entries must be in increasing order. The Y-VALUE extracted from the Table is that value associated with the X-VALUE that is equal to or less than the X-VALUE given in OPERAND 1. For example, if an X-VALUE of 3.5 is given in OPERAND 1 and the nearest X-VALUES in the Table are 3 and 4, the Y-VALUE associated with the X-VALUE of 3 will be extracted and placed in the given SET VARIABLE name. If OPERAND 1 is less than the smallest X-VALUE, the value returned is the Y-VALUE associated with the largest X-VALUE.

3.11.2 Continuous Tables

If the Table accessed is continuous (TYPE is C), all X-VALUE and Y-VALUE entries must be numeric. The SET VARIABLE of EVAL is set by the following rules :

a. the Y-VALUE associated with the X-VALUE that equals OPERAND 1, or

b. the interpolation of the Y-VALUE associated with the X-VALUE which is less than OPERAND 1 and the X-VALUE greater-than OPERAND 1, or

c. the Y-VALUE associated with the largest X-VALUE, if no interpolation is possible.

3.11.3 Alphanumeric Tables

If the Table is defined as alphanumeric (TYPE is A), one or both X-VALUE and Y-VALUE for each entry must be a name of a model entity. The SET VARIABLE is set to the Y-VALUE corresponding to the X-VALUE.

If OPERAND 1 in the EVAL Primitive does not correspond to an X-VALUE in the Table referenced, an execution error message will be printed in the analyze report and the value of the SET VARIABLE will remain unchanged.

The form for the Table entity is shown in figure 3-39.

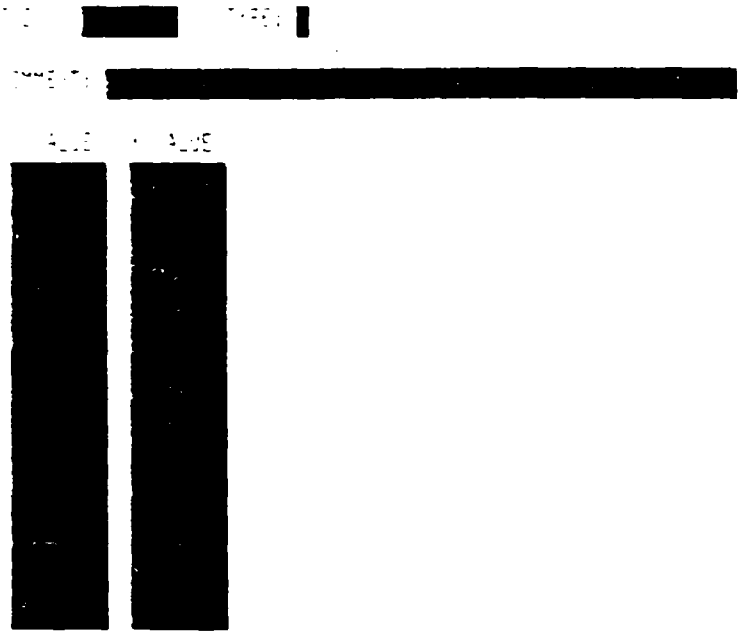


Figure 3-39. Form for the Table Entity

Following is a description of the fields in the Table form:

TABLE:	1 to 8 character name of table
TYPE:	C - continuous, D - discrete or A - alphanumeric.
X VALUE:	x-axis value
Y VALUE:	y-axis value
COMMENT:	any user comment. (0 to 53 characters)

Table entities are entered using the Design User Interface EDIT command (see section 6.1.4).

3.12 ATTRIBUTES

Certain AISIM constructs have associated attributes which can take as values, (1) numerics, (2) alpha literals (3) entity names, or (4) keywords. Some attributes are user defined. Others are dynamic attributes which are recognized and modified by the AISIM simulator.

The values of attributes may be accessed by a Process with the ASSIGN and COMPARE Primitives. The forms for both of these Primitives use two fields to indicate the value accessed. The first field contains the name of the entity and the second the name of an attribute associated with it.

Three AISIM entities, Processes, Resources and Items, may have attributes specified by the user. These attributes allow the modeler to define a unique set of characteristics for certain entities. An example is a channel. Channels have a physical attribute of maximum transfer rate. This characteristic is assigned to the AISIM Resource by specifying an attribute of RATE for the channel Resource.

Simulation experience has shown that some logic in a system is dependent on the system's dynamics. That is, some activity is dependent on queue lengths or the number of busy Resources. Since this phenomenon is fairly common, AISIM has embedded features to model this. The following attributes are built into the AISIM simulator for each instance of an entity. These attributes may be accessed by the COMPARE and ASSIGN Primitives, but the values for the Resource and Queue attributes may not be changed by the user.

Entity	Attribute	Description
Resource	NIDLEQ	the number of units of the Resource which are in an idle state
	NBUSYQ	the number of units of the Resource which are in a busy state
	NINACTQ	the number of units of the Resource which are in an inactive state
	NWAITQ	the number of Processes executing which are waiting for a Resource unit to be deallocated
Item	TAIL	the sequential creation number of the Item
	PRIORITY	the priority of the Item
Queue	NQUEUE	the number in the Queue
	TQUEUE	the average time entities are in the Queue

3.13 CONSTANTS AND GLOBAL VARIABLES

Constants and Variables are entities used to define global parameters of a model, that is, values which may be accessed by all Processes. There is an implicit caution which must be used when using these entities. Because AISIM simulates multi-processing, global parameters can be accessed "concurrently" by more than one Process. Care should be taken when multiple Processes modify the same global Variable.

A Constant is given a numeric value before the start of a simulation. The value must be numeric and can not be changed by the simulation. A Variable may be set to (1) an alpha literal, (2) the value of a keyword, or (3) to any other AISIM entity that may be accessed by the EVAL and ASSIGN Primitives. A Variable's value may vary throughout the simulation.

The initial values of both Constants and Variables are set in the Design portion of AISIM. The value of both entities may be reset, before the simulation is started, in the Analysis function.

While the value of a Constant may not be changed during the simulation, the initial value of a Variable may be changed by the user (between periods or at break points) or by the model itself (by use of the ASSIGN and EVAL Primitives).

Constants and Variables may be used in place of a numeric value anywhere a numeric value is required with the following exceptions:

1. The number of units of a Resource may only be a Constant or a numeric value.

2. The initial value of a Constant must be a numeric value.

The forms for Constants and Variables are shown in figure 3-40.

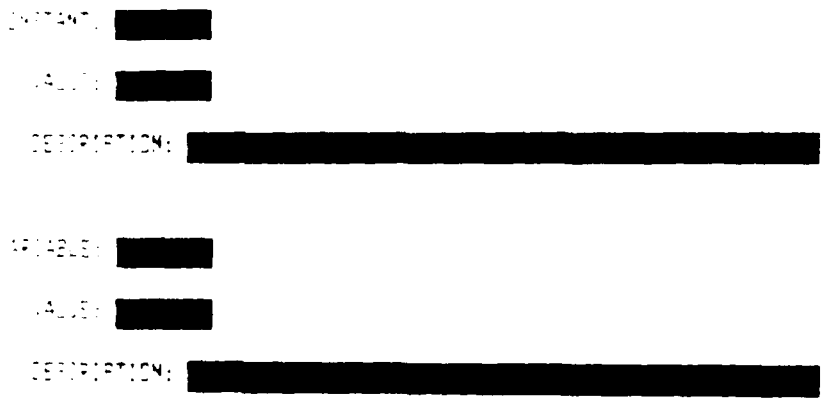


Figure 3-40. Forms for Constant and Variable Entities

Following is a description of the fields in the Constant and Variable forms:

VARIABLE/CONSTANT:	1 to 8 character name of Variable or Constant.
VALUE:	8 digit floating point or any AISIM variable reference to a numberic value.
DESCRIPTION:	Any user comment. (0 to 53 characters)

Constant and Variable entities are defined using the Design User Interface *EDIT command (see section 6.1.4).*

3.14 LOCAL VARIABLES

AISIM has two kinds of variables: local and global. Global Variables are those explicitly defined for the model and given initial values. Local variables are ones that appear in Process Primitives but are not otherwise defined. Local variables enable Processes to execute in parallel without interfering with each other because each Process has an independent set.

At the beginning of the execution of a Process all local variables are initialized to zero. They will remain so unless other values are explicitly assigned to them. Local variables may be assigned values with the ASSIGN and EVAL Primitives or through parameter passing. Local variables may be assigned the following values:

Numeric - a floating point or integer number

Global Constant or Global Variable value

Another local variable

A Resource name

A Process name

An Item name

A Queue name

An alpha literal (first character $)

The value of a keyword evaluation

Although "local," the values of such variables can be communicated from one Process to another through parameter passing (i.e., through the CALL Primitive). Local variables can be used to fill in any parameter slot in any Primitive that is not an option, a label, a distribution or function, and including:

Item attribute

Resource attribute

Process attribute

CALL given parameter

CALL return parameter

Process given parameter

Process return parameter

ALLOC Resource name

DEALLOC Resource name

CALL Process name

CALL Priority name

ASSIGN set variable (variable 2)

COMPARE variable

FILE Queue name

FILE Item name

FIND Queue name

FIND Item name

REMOVE Queue name

REMOVE Item name

RESUME task reference

3.15 ALPHA LITERALS

An alpha literal is a character string. It consists of a $ followed by up to seven other characters, as in

```
$WAIT
```

and

```
$JONES
```

that do not make up the name of a keyword (see next section). Alpha literals can be used to compare strings for identity or nonidentity with the COMPARE Primitive. They can be used as attributes. This is useful for making AISIM models more readable.

3.16 KEYWORDS

The following keywords are defined in the AISIM simulator and may be used in Process logic in any Primitive in which the evaluation of the keyword results in a value which is correct in context.

Like alpha literals, these terms begin with the character "$". However, keywords function differently from alpha literals. Keywords evaluate to a value. In that sense they can be considered intrinsic functions.

$CLOCK - The value of the current simulation clock during the execution of a simulation run. This keyword may be placed in any field of a Process Primitive which may contain a numerical value.

$CNODE - The reference to the current node in which a Process is executing. All Processes can be set to execute in a node in the architecture. The node corresponds to a Resource. This keyword evaluates to the Resource. This keyword allows a modeler to control allocation and deallocation of a node from within the execution of a Process. This keyword can be assigned a value. This, in effect, changes the node in which a Process is logically executing. This is the only keyword that may be assigned a value in the Process logic.

$TASK - The current instance of the Process in which this keyword appears. A Process executing in a simulation can assign the value of the $TASK keyword to a global Variable. This allows one Process to suspend itself and another Process to resume it by referencing the Process to be resumed with the stored value of $TASK.

$PRIORTY - The priority of the currently executing Process. This keyword is generally used in an ALLOC Primitive to resolve Resource contention issues.

$NODE - $NODE takes one argument, a reference to a Process. Given a Process, $NODE evaluates to the name of the node in which the Process has been defined to execute. This is the name of a Resource. This keyword allows a Process in AISIM to determine a destination for messages which request a specific Process to be executed. The node specification for a Process is defined by a user and is associated with the START symbol for the Process.

The following keywords directly access the legal path table and architecture structure. Each keyword evaluates to the name of a node or link Resource.

$NXTNODE - $NXTNODE takes one argument, a reference to a destination node. Given a destination node, $NXTNODE assumes the current node ($CNODE) of the executing Process is the source (FROM) node. Accessing the legal path table, $NXTNODE returns the name of the next node along the path to the destination node. This is the name of a Resource. This keyword allows the AISIM modeler to write Processes that perform message forwarding through a network.

$LINK - $LINK takes one argument, a reference to a destination node. Given a destination node, $LINK assumes the current node ($CNODE) of the Process is the source (FROM) node. Accessing the legal path table, $LINK evaluates to the name of the link to the next node along the path to the destination node. This is the name of a Resource.

3.17 MESSAGE ROUTING SUBMODEL

When one Process triggers another through a CALL Primitive, the called Process is initiated in the same node as the calling Process. This is implicit in the AISIM simulator and is true even if the called Process is associated with a different node.

In order to model the functional distribution of Processes throughout a network, a logical Process communication feature had to be incorporated into AISIM. One requirement for this feature is that the delays inherent in the network communications be accurately represented in the model so that if a Process resident in one node initiates a Process resident in another node, the delays and queueing effecting this communication are taken into account. Also, AISIM is required to enable the analysis of different architectures performing the same functions with a minimum of change to the model.

To satisfy these requirements a special submodel has been devised to represent the routing of messages through an AISIM architecture and to initiate remote Process triggering. Since different protocols for network communication are conceivable, the AISIM message routing function has been implemented as an AISIM model and included in the AISIM system library under the name COMMUN-B. This enables an AISIM user to select and merge this model into his own. The advantage of this approach is that the user can review the logic in this submodel, determine its appropriateness to his problem and modify the message routing submodel if necessary. This will not often be the case because the message routing submodel applies to many communications networks.

The message routing submodel uses the architecture and Legal Path Table of a model through the use of the system-defined keywords and the Process Primitives.

The message routing submodel consists of one Item representing the message dispatched through the system architecture, four Processes representing the activities required for the inter-node communication and other supporting entities. Everything required for this model is included in the AISIM system library and can be merged into a user's model in a simple operation. (See section 10.2 of the Library User Interface.)

Additional details on the message routing submodel are provided in appendix D.

SECTION 4

AISIM SYSTEM OVERVIEW AND SYSTEM INITIALIZATION

The AISIM user interface consists of the following levels of operation:

Level 1 - Not connected level
Level 2 - VAX/VMS Ready level
Level 3 - AISIM READY level
Level 4 - Level 4A - Design User Interface (DUI) Sublevel
Level 4B - Analysis User Interface (AUI) Sublevel
Level 4C - Replot User Interface (RUI) Sublevel
Level 4D - Hardcopy User Interface (HUI) Sublevel
Level 4E - Library User Interface (LUI) Sublevel
Level 5 - Level 5A1 - Process Editor Interface (PEI) Sublevel
Level 5A2 - Architecture Design Editor (ADE) Sublevel
Level 5E1 - Mergein (MI)
Level 5E2 - Mergeout (MO)
Level 5E3 - Checkin (CI)
Level 5E4 - Checkout (CO)
Level 5E5 - Convert (CONV)

The relationship of these different levels is shown in figure 4-1. The current level of operation determines the system's response to a given command. For example, the command EDIT LOAD is valid only in the DUI level. Each level prompts the user for input with a specific symbol or phrase. For example, the AISIM READY level prompts with the phrase "AISIM READY" on the screen when it expects a command to be entered from the keyboard. The DUI level, on the other hand, prompts with an "*". The prompt for each level is shown in the figure in its box. The commands used to go from one level to another are shown next to the arrows indicating the direction of transfer.

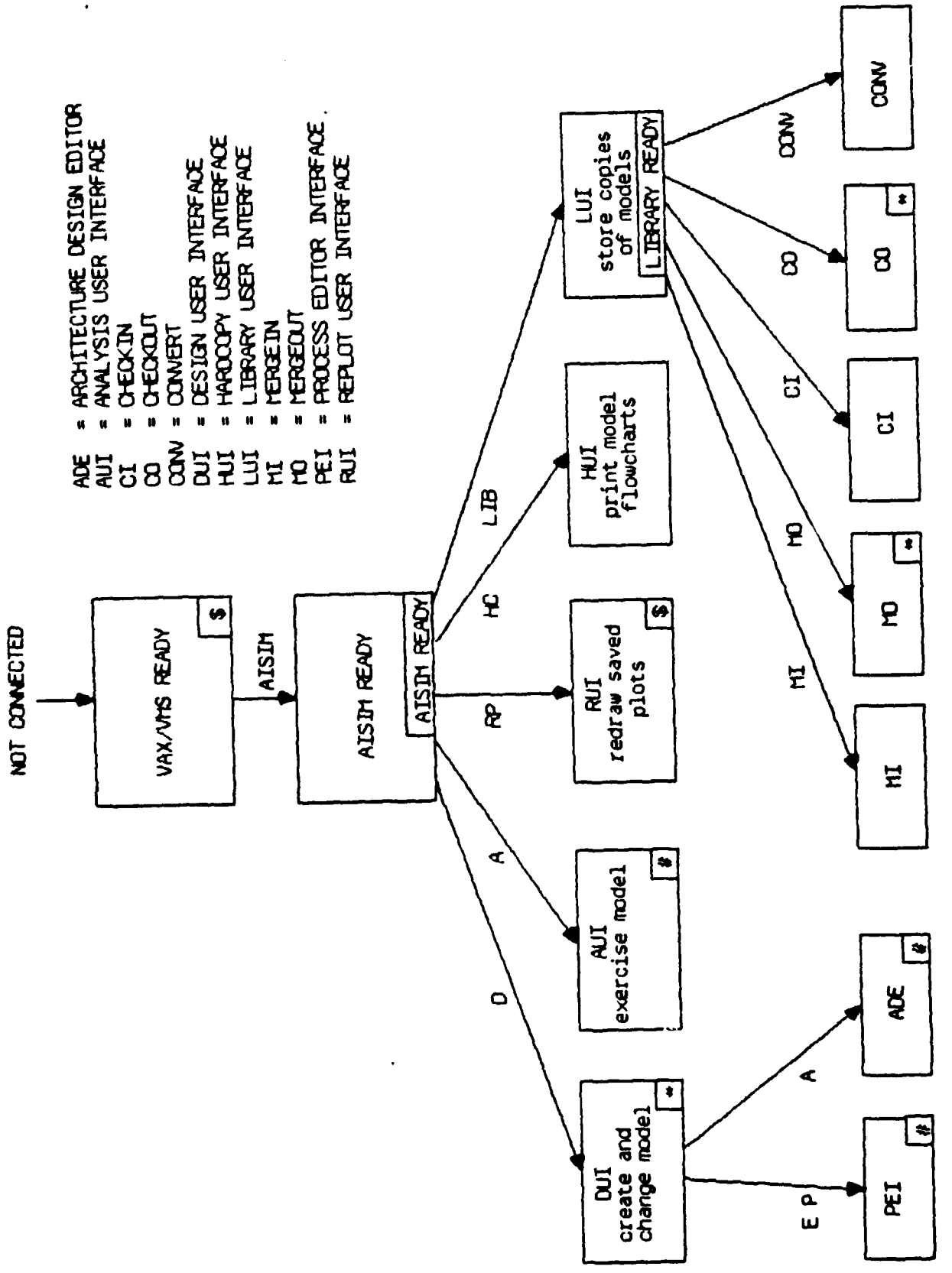


Figure 4-1. AISIM Levels of Operation

4.1 REACHING THE AISIM READY LEVEL

The procedure for logging on is specific to a given computer system and the user is referred to local references for gaining access to the top level of the system on which AISIM is hosted. (This section assumes a VAX compatible host. For other installations please refer to installation specific instructions.) When prompted with:

```
$
```

the user has reached Level 2 of AISIM operation. To reach Level 3, the user enters the command:

```
AISIM
```

When execution of this command completes, an audible 'beep' will be heard at the terminal and the AISIM READY prompt will appear on the terminal.

SECTION 5

AISIM READY LEVEL

At the AISIM READY level of operation a number of commands are available to the user for directing the course of the session (ANALYZE, CHANGE, DESIGN, END), for manipulating the database (BACKUP, EDIT, RESTORE, CONVERT), for requesting information about AISIM operation (HELP), for requesting model data (PRINT, HCOPY, GENLIST) and for deleting temporary AISIM files (DELFILE). These commands are summarized in the command summary in figure 5-1 and described in the following sections. These commands may be entered only while in the AISIM READY level of operation (i.e., when the user has received an AISIM READY prompt).

```
ANALYZE [PROJECT(project)] [NOXLATE] [TERM(terminal)]
A       [P(project)]       [N]       [T(terminal)]

BACKUP [PROJECT(project)]
       [P(project)]

CHANGE [PROJECT(project)]  [TERM(terminal)]
C      [P(project)]        [T(terminal)]

DELFILE [PROJECT(PROJECT)]
DELF    [P(project)]

DESIGN [PROJECT(project)]  [TERM(terminal)]
D      [P(project)]        [T(terminal)]

EDIT   [PROJECT(project)]  [TRACE]
       [P(project)]

END

GENLIST [PROJECT(project)] [TERM(terminal)]
GLIST   [P(project)]       [T(terminal)]

HCOPY [PROJECT(project)]   [TERM(terminal)]
HC    [P(project)]         [T(terminal)]

HELP

LIBRARY
LIB

LIST
L

LISTOFF

LISTON

MSGOFF

MSGON

PRINT [PRINT(project)]
P     [P(project)]

REPLOT [PROJECT(project)]  [TERM(terminal)]
RP     [P(project)]        [T(terminal)]

RESTORE [PROJECT(project)]
        [P(project)]
```

Figure 5-1. AISIM READY Level Command Summary

5.1 INITIATING AN ANALYSIS SESSION

Simulation of the model developed under the *DUI* sublevel (see section 5.5) is accomplished through commands available in the AUI sublevel. The AUI is accessed from the AISIM READY level by issuing the following command:

ANALYZE [PROJECT(project)] [NOXLATE] [TERM(terminal)]

A [P(project)] [N] [T(terminal)]

where:

[PROJECT(project)] is an optional parameter indicating the project database to be used. If omitted, the project is assumed to be the last project specified in a previous AISIM READY level command.

[NOXLATE] is an optional parameter indicating that, FOR THIS ANALYSIS SESSION ONLY, no translation from the "project" database is to be performed, and simulation input from a previous translation is to be used. The "previous translation" must have been performed. If this parameter is omitted, the translation will be performed.

[TERM(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

HP - HP2647A or HP2648A terminal
HP23 - HP2623 terminal
TEK - TEK4105 terminal
VT - VT100 terminal with Selanar graphics

The system will respond with the following:

```
CURRENT PARAMETERS IN EFFECT:
VERSION:     PRODUCTION VERSION 4.0
TERMINAL:    Terminal type specified in command or default
PROJECT:     Project specified in command or default
USER:        Userid
XLATE/NOXLATE:  XLATE/NOXLATE, depending upon command.
ENTER YES TO PROCEED, NO TO ABORT...
```

Typing yes will cause the system to complete the transfer to the AUI sublevel. A 'beep' will be given at the terminal and the AUI prompt (#) will appear when the system is ready to accept commands at the AUI sublevel. These commands are discussed in section 7.

During an Analysis session, various files are created. The translator creates a file called project.XLT. This file is a formatted file containing all of the entity data from the project data base, and it is used as input to the simulator. An analysis data base called project.PLT is created to hold any plot data generated by the simulation which the

user wishes to save. This file is not created if a copy already exists. The simulation report is stored in a file called project.RPT. Any trace output is stored in a file called project.TRC. All of the above files remain at the end of a simulation run. Three temporary files, PLOTDEF.DAT, PLOTDATA.DAT and SAVEPLOT.DAT, are used during an Analysis session to store plot data and definitions, and they are deleted when the simulation completes.

5.2 BACKING UP A DATABASE

To provide a backup of a project database, especially useful for saving a copy of the present model design before it is altered or modified, enter the following command:

```
BACKUP [PROJECT(project)]

BACKUP [P(project)]
```

where:

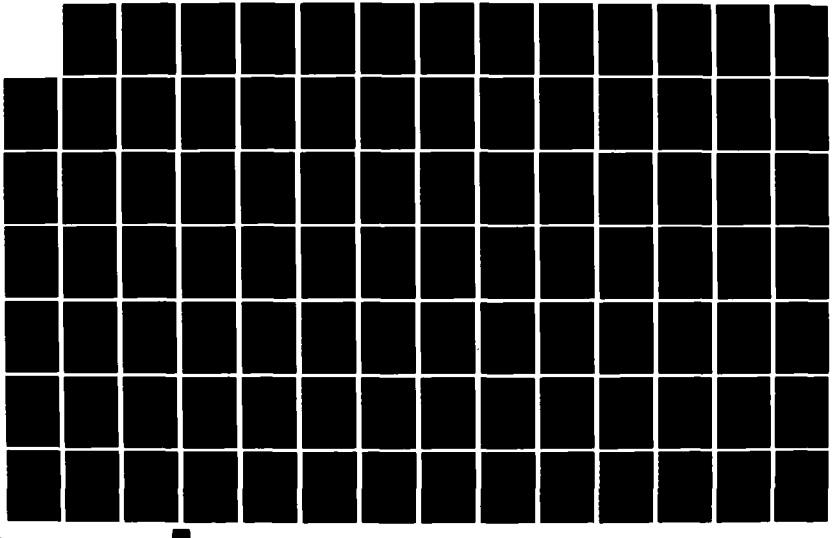
[PROJECT(project)] is an optional Parameter indicating the project database to be backed up. If omitted, the project is assumed to be the last project specified in a previous AISIM READY level command.

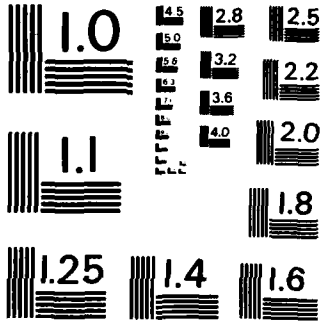
The system responds with the following:

```
CURRENT PARAMETERS IN EFFECT:
VERSION:    PRODUCTION VERSION 4.0
TERMINAL:   Default terminal type
PROJECT:    Project specified in command or default
USER:       Userid
ENTER YES TO PROCEED, NO TO ABORT...
```

A "yes" response will cause a backup copy of the project to be stored in a database file named project.BCK. A "no" response will abort the command.

AD-A161 556 ASIM (AUTOMATED INTERACTIVE SIMULATION MODELING SYSTEM) 2/4
VAX VERSION USER'.. (U) HUGHES AIRCRAFT CO FULLERTON CA
GROUND SYSTEMS GROUP S KNEEBURG FEB 85 ESD-TR-85-127
UNCLASSIFIED F33615-81-C-5098 F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

5.3 CHANGING THE CURRENT PARAMETERS

The current parameters of an AISIM session (PROJECT and TERMINAL) can be changed via the CHANGE command. The syntax for the CHANGE command is as follows:

```
CHANGE [PROJECT(project)] [TERM(terminal)]

C [P(project)] [T(terminal)]
```

where:

[PROJECT(project)] is an optional parameter indicating the new project database to be used. If omitted, the project default value remains unchanged.

[TERM(terminal)] is an optional parameter indicating the new terminal type to be used. If omitted, the terminal type default value remains unchanged. The valid terminal types are the following:

```
HP   - HP2647A or HP2648A terminal
HP23 - HP2623 terminal
TEK  - TEK4105 terminal
VT   - VT100 terminal with Selanar graphics
```

This command causes the current default project and terminal to be set to the names entered. The current default parameters are then listed as follows:

```
CURRENT PARAMETERS IN EFFECT:
VERSION:   PRODUCTION VERSION 4.0
TERMINAL:  Default terminal type
PROJECT:   Default project
USER:      Userid
```

5.4 DELETING PROJECT FILES

The DELFILE command is used to delete the following five files for a specified project:

1) project.XLT

2) project.WDB

3) project.RPT

4) project.LST

5) project.TRC

To delete these files, the user types:

```
DELFILE [PROJECT(project)]

DELF    [P(project)]
```

where:

[PROJECT(project)] is an optional parameter specifying the project name for the files. If omitted, the project is assumed to be the last project specified in a previous AISIM READY level command.

The system responds with the following:

```
CURRENT PARAMETERS IN EFFECT:
VERSION:  PRODUCTION VERSION 4.0
TERMINAL: Terminal type default
PROJECT:  Project specified in command or default
USER:     Userid
ENTER YES TO PROCEED, NO TO ABORT...
```

A "yes" response will cause the files to be deleted. A "no" response will abort the command.

5.5 INITIATING A DESIGN SESSION

A project database is created/modified using the commands available in the DUI. The DUI is accessed from the AISIM READY level by issuing the following command:

DESIGN [PROJECT(project)] [TERM(terminal)]

D [P(project)] [T(terminal)]

where:

[PROJECT(project)] is an optional parameter indicating that the desired project file to be acted upon by the command is "project", where "project" is a standard alphanumeric file label containing 1-8 characters beginning with an alpha character and containing no special characters or imbedded blanks.

[TERM(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

HP - HP2647A or HP2648A terminal
HP23 - HP2623 terminal
TEK - TEK4105 terminal
VT - VT100 terminal with Selanar graphics

The following is displayed after entering this command:

```
CURRENT PARAMETERS IN EFFECT:
VERSION:  PRODUCTION VERSION 4.0
TERMINAL: Terminal type specified in the command or default
PROJECT:  Project specified in the command or default
USER:     Userid
ENTER YES TO PROCEED, NO TO ABORT...
```

Typing YES causes the completion of the level transfer. The terminal will display:

CREATING WORKING DATABASE.....

Followed by:

.....COPY COMPLETE

The DUI prompt (*) will appear when the system is ready to accept commands at the DUI sublevel. These commands are discussed in section 6.

The project database is stored in a database file named project.DBF. The working copy of the database is stored in a database file named project.WDB.

5.6 VIEWING OUTPUT REPORTS

To access the model simulation report or model trace interactively on the terminal (via the EDT editor), enter the following command:

```
EDIT [PROJECT(project)]

EDIT [P(project)]
```

or

```
EDIT [PROJECT(project)] [TRACE]

EDIT [P(project)] [TRACE]
```

Result:

The EDT editor is entered with the file to be edited set according to the project. All EDT editor commands can be used on this file. The file is either the project report file (this is the default) or the project trace file. See section 11.3 for a brief discussion of relevant EDT text editor commands.

5.7 RETURNING TO VAX/VMS READY LEVEL

To return to the VAX/VMS Ready level from the AISIM READY level, the user types the command:

```
END
```

The system will return to the VAX/VMS Ready Level and the screen will display

```
$
```

5.8 CREATING A MODEL LISTING

The GENLIST command is used to produce a listing of a model without having to enter the AUI level and perform a complete translation of the model. The listing is identical to the Initialization Report section of the output report (see the section on AISIM Simulation Results Reporting). Elements of this report are:

1) Global Constant Definition
2) Table Definition
3) Global Variable Definition
4) Item Definition
5) Queue Definition
6) Resource Definition
7) Architecture Legal Path Definition
8) Action Definition
9) Process Definition
10) Load Definition
11) Scenario Definition

To obtain a listing, the user types:

```
GENLIST [PROJECT(project)] [NOXLATE] [TERM(terminal)]

GLIST   [P(project)]       [N]       [T(terminal)]
```

where:

[PROJECT(project)] is an optional parameter specifying the project database for which a listing is desired. If omitted, the project is assumed to be the last project specified in a previous AISIM READY level command.

[NOXLATE] is an optional parameter indicating that the listing of the model will be from a previous translation of the model. If this parameter is omitted, a translation will be performed. (The translation listing is stored in a temporary file; the user's current translation file, if there is one, is not affected by this procedure.)

[TERM(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

```
HP   - HP2647A or HP2648A terminal
HP23 - HP2623 terminal
TEK  - TEK4105 terminal
VT   - VT100 terminal with Selanar graphics
```

The system responds with the following:

```
CURRENT PARAMETERS IN EFFECT:
VERSION:         PRODUCTION VERSION 4.0
TERMINAL:        Terminal type specified in command or default
PROJECT:         Project specified in command or default
USER:            Userid
XLATE/NOXLATE:   XLATE/NOXLATE, depending upon command
ENTER YES TO PROCEED, NO TO ABORT...
```

A "yes" response will cause the listing to be created and a copy to be automatically printed. A "no" response will abort the command.

The listing is stored in a file named project.LST.

5.9 HARDCOPY OUTPUT OF THE PROCESS FLOWCHARTS

Hardcopy graphics of Process flowcharts are obtained in the Hardcopy User Interface (HUI). The HUI is accessed from the AISIM READY level by issuing the following command:

HCOPY [PROJECT(project)] [TERM(terminal)]

HC [P(project)] [T(terminal)]

where:

[PROJECT(project)] is an optional parameter indicating the project database with the Processes of interest. If omitted, the project is assumed to be the last project specified in a previous AISIM READY level command.

[TERM(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

HP - HP2647A terminal
HP23 - HP2623 terminal
TEK - TEK4105 terminal

The system will respond with the following:

```
CURRENT PARAMETERS IN EFFECT:
VERSION:  PRODUCTION VERSION 4.0
TERMINAL: Terminal type specified in the command or default
PROJECT:  Project specified in command or default
USER:     Userid
ENTER YES TO PROCEED, NO TO ABORT...
```

A "yes" response will cause the HUI to be invoked. The system will then prompt the user for all required information (see section 9 on the HUI).

Note: This function is not available on a VT100 terminal.

5.10 OBTAINING HELP FROM THE SYSTEM

To obtain help from the system, type the following command:

```
HELP
```

The user will receive summary help information on all commands.

5.11 EXERCISING THE LIBRARY FACILITY

The Library User Interface (LUI) allows the user to do the following:

1. Move entities from a model project database into a storage area called a "buffer".
2. Move entities from a "buffer" into the database of another model project.
3. Move entities from a "buffer" into a library of model entities.
4. Move entities from a library to a "buffer".
5. Convert a pre-version 4.0 project database to a version 4.0 compatible project database

The LUI is entered by issuing the command:

```
LIBRARY

LIB
```

The system will respond with the prompt:

```
LIBRARY READY
```

and the user may invoke any of the LUI sublevels listed in the LUI Command Summary (see section 10).

16

5.12 LISTING THE CURRENT OPTIONS

To list the current options in effect, type the following command:

```
LIST
```

```
L
```

The system will display the current options in effect, including PROJECT, USER, VERSION, and TERMINAL.

5.13 LISTING THE COMMAND PROCEDURE LINES

If a user is having problems from the AISIM READY level or LIBRARY READY level which may stem from missing system files or an operating system problem, the user can set a flag so that all of the files which control the execution of an AISIM session will be displayed as they are executed. This flag is set by typing the following command:

```
LISTON
```

When this option is in effect, all VAX/VMS commands which set up an AISIM session will be displayed at a user's terminal as they are executed. Viewing the commands as they execute may help a user determine where a problem is occurring.

5.14 DISABLE THE LISTON OPTIONS

In order to disable the LISTON option, i.e., to inhibit the displaying of VAX/VMS commands as they are being executed, type the following command:

```
LISTOFF
```

This command disables the command listing mode initiated by the LISTON command.

5.15 *DISABLE AISIM MESSAGES*

Upon invoking each AISIM function, the user is presented with the current version, terminal type, project, etc., and asked if (s)he wants to continue or abort. These messages and prompt can be suppressed by typing the following command:

```
MSGOFF
```

When the user invokes a function, control will be transferred directly to that function without further prompting.

5.16 DISABLE MSGOFF FEATURE

If the user has disabled the AISIM messages and prompts via the MSGOFF command, the messages and prompts can be turned back on via the following command:

```
MSGON
```

Following this command, the user will receive the version, terminal type, project, etc. messages and prompt to continue whenever an AISIM function is invoked.

5.17 PRINTING OUTPUT REPORTS

To request printing of the model output report, type the following command:

PRINT [PROJECT(project)]

P [P(project)]

where:

PROJECT(project) is an optional parameter indicating which project's report file is to be printed. If omitted, the project is assumed to be the last project specified in a previous AISIM READY level command.

Result:

The output report (project.RPT) of a project is printed. This is a report of the standard results of a simulation run.

NOTE: The output report is automatically printed at the conclusion of an Analysis session.

5.18 INITIATING A REPLOT SESSION

The Replot User Interface (RUI) allows the user to display plots which were saved during previous Analysis sessions. The command to invoke the RUI is as follows:

```
REPLOT [PROJECT(project)] [TERM(terminal)]

R [P(project)] [T(terminal)]
```

where:

[PROJECT(project)] is an optional parameter indicating the project database used in creating the saved plots. If omitted, the project is assumed to be the last project specified in a previous AISIM READY level command.

[TERM(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

```
HP   - HP2647A or HP2648A terminal
HP23 - HP2623 terminal
TEK  - TEK4105
VT   - VT100 terminal with Selanar graphics
```

The system will respond with the following display:

```
CURRENT PARAMETERS IN EFFECT:
VERSION:  PRODUCTION VERSION 4.0
TERMINAL: Terminal type specified in command or default
PROJECT:  Project specified in command or default
USER:     Userid
ENTER YES TO PROCEED, NO TO ABORT...
```

A "yes" response will cause the system to complete the transfer to the RUI. The RUI prompt ($) will be displayed when the system is ready to accept commands at the RUI sublevel.

5.19 RESTORING A DATABASE (AFTER A CATASTROPHE HAS OCCURRED)

This command is used in conjunction with the BACKUP command. If the user was editing the original database and had issued a BACKUP command against this database, then a copy of the original database exists. The RESTORE command causes the damaged original database to be replaced with this backup copy.

To restore a previously backed-up database (only necessary if a catastrophe has occurred which altered the project database, or it is desirable to restart a model from a known configuration), enter the following command:

RESTORE [PROJECT(project)]

RESTORE [P(project)]

where:

[PROJECT(project)] is an optional parameter indicating the previously backed-up project database to be restored. If omitted, the project is assumed to be the last project specified in a previous AISIM READY level command.

The backed-up copy of the database, called project.BCK, will be copied onto the damaged database and will have the database name project.DBF.

SECTION 6

DESIGN USER INTERFACE (DUI)

The DUI and its lower levels are used to define a model by creating, modifying, or deleting AISIM model entities. The Action, Constant, Item, Load, Process, Queue, Resource, Scenario, Table, and Variable entities are created and edited at the DUI level, using the EDIT command. The Process entities which represent operations in the modeled system are created and edited at a sublevel of the DUI level called the Process Editor Interface (PEI). The PEI is invoked by issuing the EDIT command (at the DUI level) and specifying a Process as the entity to be edited. A system architecture and its related Legal Path Table, nodes, and links are defined in a second sublevel of the DUI called the Architecture Design Editor (ADE). The ADE is invoked by issuing the ARCH command at the DUI level.

When creating and editing entities in the DUI level, the system prompts the user for further information by use of forms. Each form specifies the required and optional attributes of its respective entity-type. The areas on which information is to be entered appear in "reverse video" (dark characters on a light background), and indicate the attributes that are to be supplied by the user.

Each time the user presses the keyboard carriage return key, the character cursor is positioned to the start of another designated area. The user enters parameters requested by the form by keying in the desired alphanumeric information. If the user changes his mind about the parameters previously keyed in, he may alter them by merely writing over the old information. When the user is satisfied with the contents of the form, he inputs it to the computer by exiting the form. Below is a complete description of the use of forms.

While the user is in the DUI, all changes are made to a working copy of the user's database. When the user issues a SAVE command during or at the end of the DUI session, the working database is copied back into the user's real database. This procedure enables the user to change his/her mind about changes made in the working database and to protect the user's real database in case the computer crashes during a DUI session.

The AISIM DUI commands used to input, modify, and delete entities from the model, are illustrated in figure 6-2 and described on the pages that follow it.

USE OF THE FORMS EDITOR

This section describes the use of the forms editor on the various terminals. Figure 6-1 is a chart which describes the keys used to achieve specific movements through a form. Following the figure is a description of each of the ways of moving through a form.

	UP	DOWN	LEFT	RIGHT	ENTER	+FIELD	-FIELD
HP2647A	F1	F2	F3	F4	F5	<cr>	F6
HP2648A	F1	F2	F3	F4	F5	<cr>	F6
HP2623	F1	F2	F3	F4	F5	<cr>	F6
TEK4105	F1	F2	F3	F4	F5	<cr>	F6
VT100	↑	↓	<--	-->	PF1	<cr>	PF2

Figure 6-1. Terminal Profiles

UP - If the cursor is in a block of fields, such as Resource attributes, the cursor will move up to the field above it. If the cursor is in a single field or at the top of a block, the cursor will move to the end of the next field above it. If there are no fields above it, the cursor will wrap to the end of the last field in the form.

DOWN - If the cursor is in a block of fields, such as Resource attributes, the cursor will move down to the field below it. If the cursor is in a single field or at the bottom of a block, the cursor will move to the beginning of the next field below it. If there are no fields below it, the cursor will wrap to the beginning of the first field in the form.

LEFT - The cursor will move one position to the left in the current field. If the cursor is at the beginning of a field, it will move to the end of the previous field. If the cursor is at the top of the form, it will wrap to the end of the last field in the form.

RIGHT - The cursor will move one position to the right in the current field. If the cursor is at the end of a field, it will move to the beginning of the next field. If the cursor is at the end of the form, it will wrap to the beginning of the first field in the form.

ENTER - Cut the form and send the data in the form to be processed by the AISIM system.

+FIELD - Move the cursor to the beginning of the next field in the form. If the cursor is at the end of the form, it will wrap to the top of the form.

-FIELD - Move the cursor to the end of the previous field in the form. If the cursor is at the top of the form, it will wrap to the end of the last field in the form.

The BACKSPACE key can also be used to back up and make changes in a field.

Each time a user advances from one field to the next, the terminal 'beeps' to signal the change in fields.

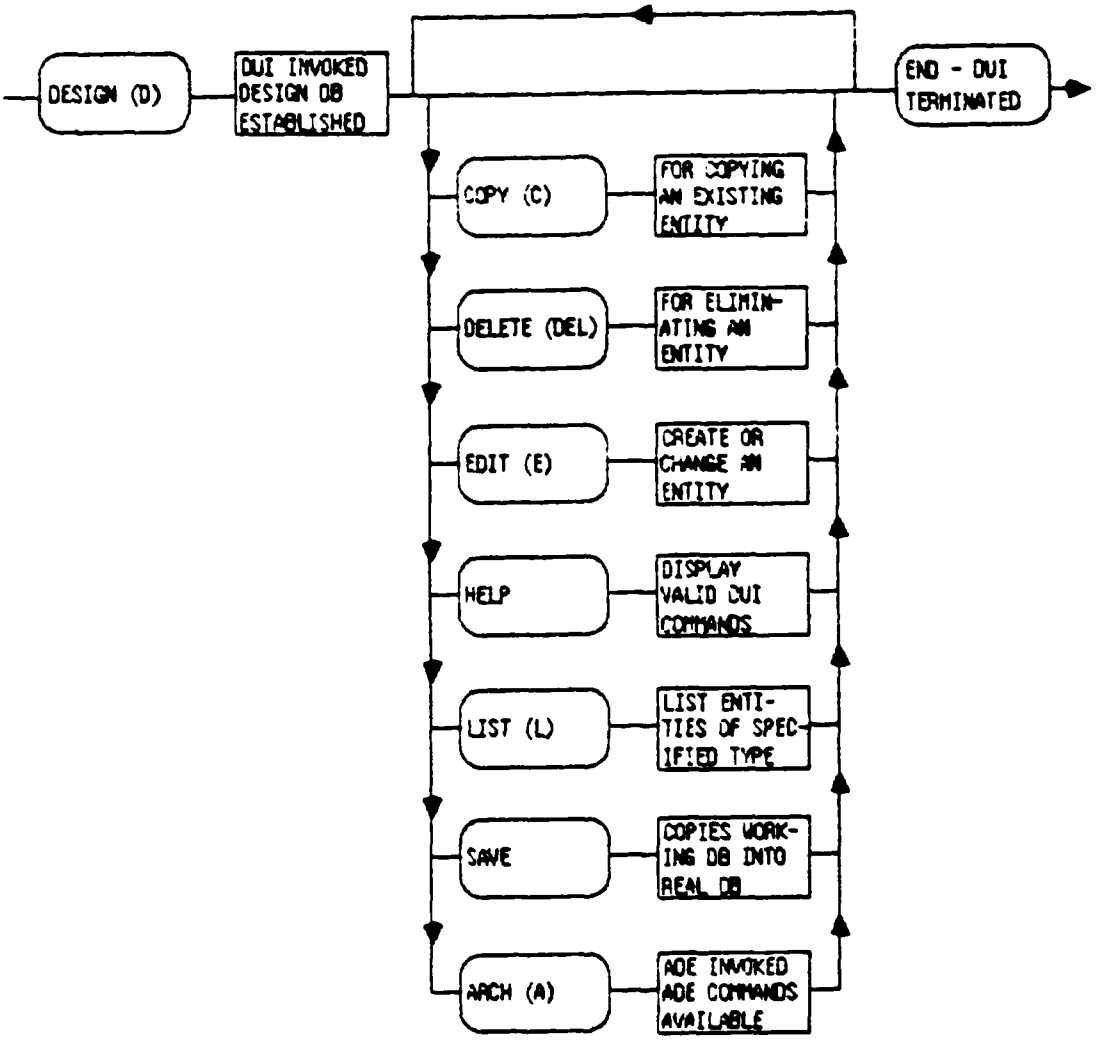


Figure 6-2. Design User Interface Commands

6.1 DUI COMMAND SUMMARY

Figure 6-3 contains a summary of the DUI level commands.

```
ARCH
A

COPY {entity-type},{existing-name},{new-name}
C

DELETE {entity-type},{entity-name}/*
DEL

EDIT {entity-type},{entity-name},[OLD/NEW]
E

END

HELP

LIST {entity-type}
L

SAVE
```

Figure 6-3. DUI Command Summary

6.1.1 DUI COMMAND: ARCH

The ARCH command is used to invoke the Architecture Design Editor (ADE).

This command is valid only in the DUI Ready Level.

COMMAND SYNTAX:

ARCH

A

FUNCTION RESULT:

The ADE is invoked so that the architecture is built under the project designated by the DESIGN command. A # prompt is provided for the user to input ADE commands. These commands are discussed in section 6.3.

6.1.2 DUI COMMAND: COPY

The COPY command is used to create a copy of an existing entity.

COMMAND SYNTAX:

COPY {entity-type},{existing-name},{new-name}

C

where:

{entity-type} is a required parameter indicating any valid entity type.

Entity-type may be any of the following:

Entity-type	Acceptable Abbreviation
Action	A
Constant	C
Item	I
Load	L
Process	P
Queue	Q
Resource	R
Scenario	S
Table	T
Variable	V

{existing-name} is a required parameter identifying the existing entity whose parameters are to be duplicated.

{new-name} is a required parameter which specifies the name of the new entity whose parameters are duplicates of the "existing entity".

If entity type, existing-name or new-name is missing or invalid, the user is prompted.

A carriage return entered in response to any prompt aborts the command and returns the user to the DUI Ready state - * prompt.

6.1.3 DUI COMMAND: DELETE

The DELETE command is used to eliminate a named entity of a given type from the user database. A restriction on the use of this command is that Resources associated with architectural nodes or links cannot be deleted outside of the Architecture Design Editor sublevel.

COMMAND SYNTAX:

DELETE {entity-type},{entity-name}

{entity-type},{entity-name},...,{entity-name}

{entity-type},*

DEL

where:

{entity-type} is a required parameter indicating any valid entity type. The valid entity types are listed in section 6.1.2.

{entity-name} is a required parameter indicating the name of the entity to be deleted. It is permissible to give a list of entity-names, of the same type, each member of which is separated by a comma.

* is a parameter used indicate all of the entities of the specified type are to be deleted.

If entity-type or entity-name is missing or invalid, the user is prompted for a valid parameter.

A carriage return in response to the prompt aborts the command, and the user is returned to the DUI Ready state - * prompt.

FUNCTION RESULT:

If the named entity is not a Resource associated with a architectural node or link, the entity will be deleted from the user's working database. If the entity is a Resource associated with a node or link, the user will be given the message:

" entity " IS ASSOCIATED WITH THE ARCH. AND CAN ONLY BE DELETED IN THE ADE

where " entity " is the name of the entity to have been deleted.

When there is more than one such Resource listed in the command to delete the user will be given the above message for each one.

6.1.4 DUI COMMAND: EDIT

The EDIT command is used either to create an entity, or to change an existing entity.

COMMAND SYNTAX:

EDIT {entity-type},{entity-name},[OLD/NEW]

E

where:

{entity-type} is a required parameter indicating any valid entity type. The vaid entity types are listed in section 6.1.2.

{entity-name} is a required parameter indicating the name of the entity to be edited.

[OLD/NEW] is an optional parameter indicating that the named entity is to be created (NEW), or that the named entity exists (OLD) and is to be changed. If the [OLD/NEW] parameter is entered incorrectly, the user is prompted for confirmation to continue the command. The default for this parameter is OLD.

FUNCTION RESULT:

If the entity-type specified is Process, the PEI level (see section 6.2) is automatically invoked. If any other valid entity type is specified, the user is presented with a form to describe that entity. The forms for the entities are shown in figures 3-1 through 3-4, 3-6 through 3-8, 3-39, and 3-40. The user must fill out the form to input the completed entity into the working database. The user is then returned to the DUI Ready state - * prompt.

6.1.5 DUI COMMAND: END

The END command is used to terminate a DUI session.

COMMAND SYNTAX:

END

FUNCTION RESULT:

The Design session is ended. The working database is closed. If a SAVE command has not been given since the last EDIT command, the user is asked if the working database is to be saved. The query is:

SAVE (Y/N)?

If the user answers "Y", the working database is saved into the real database and the session is ended. Control is passed to the AISIM READY level (level 3). If the user answers "N", the session is ended and the working database is not saved. Control is passed to the AISIM READY level (level 3). Depressing the RETURN key in response to the SAVE query aborts the END command, and returns the user to the DUI Ready level - * prompt.

6.1.6 DUI COMMAND: HELP

The HELP command lists the commands currently available to the user during a DUI session.

This command may be used any time during a DUI session.

COMMAND SYNTAX:

HELP

FUNCTION RESULT:

The acceptable commands (i.e., the ones valid at the current level) are listed.

HELP displays the following commands:

ARCH	A	COPY	C	DELETE	DEL	EDIT
E	END	LIST	L	SAVE		

6.1.7 DUI COMMAND: LIST

The LIST command displays all entities of a specified type. Included with each entity is its name and description.

COMMAND SYNTAX:

LIST {entity-type}

L

where:

{entity-type} is a required parameter indicating any valid entity type. The valid entity types are listed in section 6.1.2.

If {entity-type} is missing or invalid, the user is prompted for a valid entity type.

A carriage return entered in response to the prompt aborts the command, and the user is returned to the *DUI Ready state - * prompt.*

FUNCTION RESULT:

The user is presented with a list of all existing entities of the requested type.

6.1.8 DUI COMMAND: SAVE

The SAVE command copies the contents of the working database into the user's permanent database.

COMMAND SYNTAX:

SAVE

FUNCTION RESULT:

The real database is replaced with the contents of the working database, and the user is returned to the DUI ready state - * prompt. The command is useful when the user is defining a large system. With the SAVE command the user saves the model design up to the point at which the command is given. This protects that portion of the design from computer failures.

6.1.9 Termination of a DUI Session

As mentioned earlier, a DUI session is terminated by issuing the END command. Syntax and results are described in the preceeding section. The DUI session is ended. The working database is closed. If a SAVE command has not been given since the last EDIT command, the user is asked if the working database is to be saved. The query is:

SAVE (Y/N)?

If the user answers "Y", the working database is saved into the real database and the session is ended. If the user answers "N", the session is ended and the working database is not saved. Depressing the RETURN key in response to the SAVE query aborts the END command, and returns the user to the DUI Ready state. When the SAVE query is answered, control is returned to the AISIM READY level and the AISIM READY prompt is displayed.

6.2 Process Editor Interface (PEI)

The PEI, coupled with the capabilities of the graphic terminal, allows the user to describe graphically the logical flow of an operation which he wishes to model. The PEI is used to build Processes which model man-machine interaction as well as data processing functions (software logic). A Process is composed of Primitives which are symbols that represent the individual steps in an operation. Using the PEI commands which are described below the user arranges the Primitives in an order that describes the Process.

6.2.1 Use of the PEI

The system transfers control from the DUI to the PEI when the user issues the following command:

EDIT PROCESS,{entity-name},[OLD/NEW]

E P,{entity-name},[OLD/NEW]

where:

{entity-name} is required parameter indicating the name of the Process to be edited.

[OLD/NEW] is an optional parameter indicating that the named entity is to be created (NEW) or that the named entity exists (OLD) and is to be changed. If the (OLD/NEW) parameter is entered incorrectly, the user is prompted for confirmation to continue. The default for this parameter is OLD.

When the PEI is entered, the screen is blanked. If a Process has already been created, the first screen of Primitives is displayed from the START symbol down. If the Process is new, a form is displayed which requests information about the Process (see section 3.8). The user must complete the form to input the PROCESS into the database. The form is cleared from the screen and the START and END Primitives of the new Process are displayed on the screen. At this point, a pound sign (#) prompt will be displayed indicating that the user may issue any of the PEI commands. The PEI commands, which are described on the following pages, are used to select, position, and describe the Primitives to create a Process. A sample Process is shown in figure 6-5 (section 6.2.10). In the following command descriptions, "position" refers to the numbers appearing at the left side of the figure.

There are two modes in the PEI: DRAW and NODRAW. Under DRAW mode, all changes to Primitives on the screen are reflected in the display. Under NODRAW mode, changes are not reflected in the display until the user explicitly requests that the display be updated. When the user first enters the PEI, DRAW mode is the default. If the user changes the mode, the change will stay in effect for all subsequent uses of the PEI until changed by the user or the user exits the DUI. These modes are explained more fully in the PEI DRAW and NODRAW commands (sections 6.2.6 and 6.2.11).

```
BOTTOM
B

Change {position}
C

DELETE {first position},[number of consecutive positions]
DEL

DOWN [number of positions]
D

DRAW
DR

END
E

HELP

HOLD {position}
H

MENU
M

NODRAW
N

PLACE {Primitive},[position]
P

REDRAW
RED

TOP
T

UP [number of positions]
U
```

Figure 6-4. PEI Command Summary

6.2.2 PEI COMMAND: BOTTOM

The Bottom command is used to display the last six Primitives in the current Process structure.

COMMAND SYNTAX:

BOTTOM

B

FUNCTION RESULT:

The bottom of the Process structure being edited is drawn from the END symbol up. The END symbol is always the last position of a Process structure.

6.2.3 PEI COMMAND: CHANGE

The CHANGE command is used to modify the user defined parameters of a Primitive within the current Process structure.

COMMAND SYNTAX:

CHANGE {position}

C

where:

{position} is a required parameter indicating the position of the Primitive, within the Process structure, whose parameters are to be changed.

FUNCTION RESULT:

When the CHANGE command is invoked, the user is presented with a form corresponding to the Primitive at the indicated position. The user may change any or none of the attributes of the Primitive. If the user is in DRAW mode and the Primitive being changed is on the screen, the Process structure is then redisplayed with any changes made; otherwise, the screen remains unchanged.

6.2.4 PEI COMMAND: DELETE

The DELETE command allows the user to delete a single Primitive, or a range of Primitives, from the current Process structure.

COMMAND SYNTAX:

DELETE {first position},[number of consecutive positions]

DEL

where:

{first position} is a required parameter indicating the position of the first Primitive to be deleted.

[number of consecutive positions] is an optional parameter indicating the number of consecutive positions to be deleted, starting with the Primitive indicated by the {first position} parameter. If this parameter is omitted, the default condition is to delete only the Primitive at the position indicated by the {first position} parameter.

FUNCTION RESULT:

The Primitives indicated by the {first position} parameter and the optional parameter are deleted from the Process structure. The START and the END symbols may not be deleted. Additionally, the numbers of all Primitives being deleted must be displayed on the screen.

If the user is in DRAW mode, this simply means that the Primitives to be deleted must be visible. If the user specifies to delete Primitives past the end of the screen, only the Primitives on screen will be deleted. After the delete command is issued, the remaining Primitives in the stucture are scrolled up.

If the user is in NODRAW mode, the numbers of the primitives being deleted must be on screen, but not necessarily the symbols themselves. For example, say the first six Primitives are being displayed and the user deletes Primitives three through six. Since the legend still shows three through six, the user can delete the new third through sixth Primitives even though the symbols on screen may not correspond to the Primitives being deleted. The user should take care when deleting Primitives while in NODRAW mode. If the user specifies to delete Primitives whose numbers are past the end of the screen, only the Primitives whose numbers are on screen will be deleted.

6.2.5 PEI COMMAND: DOWN

The DOWN command allows the user to "jump down" the current Process structure an indicated number of positions.

COMMAND SYNTAX:

DOWN [number of positions]

D

where:

[number of positions] is an optional parameter indicating the number of positions that the structure is to "jump down". If this parameter is not used, the default condition is to drop the Process structure down six Primitives, which is analogous to displaying the next page.

FUNCTION RESULT:

The Process structure jumps down the number of positions indicated by the optional parameter, if given. Otherwise the structure jumps down six Primitives or to the bottom of the structure if less than six Primitives follow the last position currently displayed.

6.2.6 PEI COMMAND: DRAW

The DRAW command is used to put the user in the PEI DRAW mode.

COMMAND SYNTAX:

DRAW

DR

FUNCTION RESULT:

The DRAW command sets the PEI mode to DRAW mode. This mode will remain in effect for all future PEI sessions during the current DUI session until changed by a NODRAW command.

In DRAW mode, all changes made by a user to the portion of the Process structure currently being displayed will be reflected in the display. I.e., if a Primitive is changed via the CHANGE command, that Primitive will be redrawn on the screen. If Primitives on the screen are deleted, remaining Primitives will be scrolled up to fill the display.

If changes are made to the Process in an area which is not currently being displayed, those changes will not be reflected in the display until the user redraws the area in which the changes were made. For example, if the user places new Primitives at the bottom of the Process, and the bottom is off the screen, the new Primitives will not be displayed until the user explicitly displays the bottom of the Process structure.

6.2.7 PEI COMMAND: END

The END command is used to terminate and exit the PEI session.

COMMAND SYNTAX:

END

E

FUNCTION RESULT:

The PEI session is ended, the graphics display is erased, and the user is returned to the DUI Level.

6.2.8 PEI COMMAND: HELP

The HELP subcommand displays a list of the valid PEI commands.

COMMAND SYNTAX:

HELP

FUNCTION RESULT:

The list of valid commands (see PEI Command summary, figure 6-4) is displayed.

6.2.9 PEI COMMAND: HOLD

The HOLD command allows the user to insert any valid Primitive, which is already a part of the current Process structure, into the menu item "HOLD" so that it may be replicated.

COMMAND SYNTAX:

HOLD {position}

H

where:

{position} is a required parameter indicating the position of the Primitive which is to be placed in hold for the purpose of replication.

FUNCTION RESULT:

The Primitive (complete with the previously defined parameters) is placed in hold. This item may then be replicated by using the PLACE command and using HOLD as the Primitive to be placed. When a Primitive is stored in Hold, it remains there, accessible to the user throughout the DUI session, and thus Primitives may be moved from one Process to another. When there is a Primitive in hold on a terminal on which the menu can be displayed (see MENU command), the name of the Primitive being held appears below the menu display area preceded by an asterisk (example: *CREATE).

6.2.10 PEI COMMAND: MENU

MENU is used to display the possible Primitives for a Process.

COMMAND SYNTAX:

MENU

M

FUNCTION RESULT:

The menu is a one-column list of names of the valid Primitives (see section 3.9 for a description of the Primitives). If the menu will fit on the screen, it is displayed to the left of the Process flowchart. If the menu will not fit on the screen, a message will be displayed noting that fact. The menu can be displayed on HP2647A, HP2648A, and TEK4105 terminals. Figure 6-5 shows the Process menu.

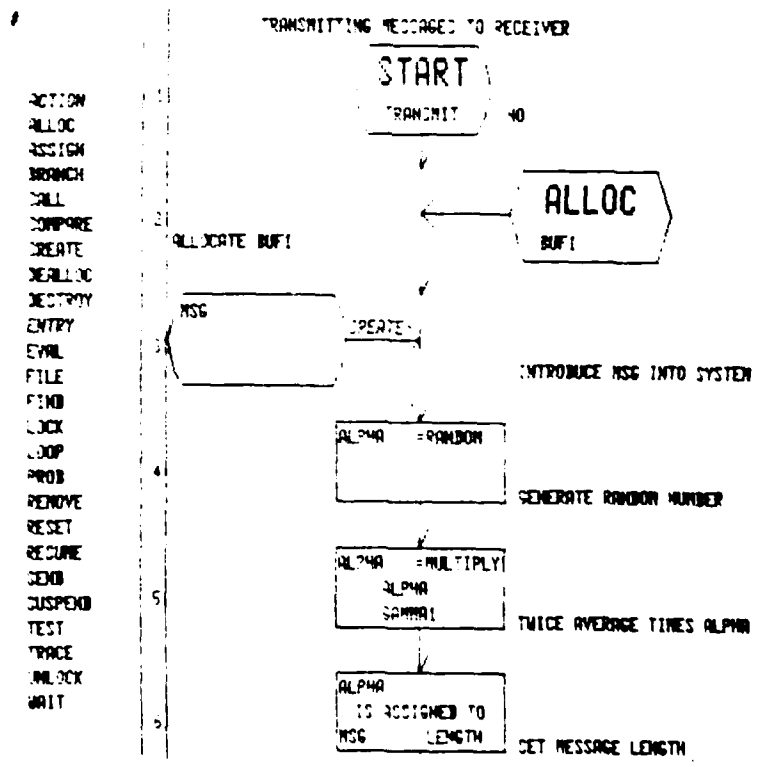


Figure 6-5. Process Display with Menu

6.2.11 PEI COMMAND: NODRAW

The NODRAW command is used to put the user in the PEI NODRAW mode.

COMMAND SYNTAX:

NODRAW

N

FUNCTION RESULT:

The NODRAW command sets the PEI mode to NODRAW mode. This mode will remain in effect for all future PEI sessions during the current DUI session until changed by a DRAW command.

In NODRAW mode, no changes which are made to Primitives in the Process are reflected in the display until the display is explicitly redrawn by the user. Commands which can be used to update the display are TOP, BOTTOM, UP, DOWN and REDRAW. The user should take care when deleting Primitives while in NODRAW mode to guard against deleting necessary Primitives since the screen is not updated after a DELETE is performed.

6.2.12 PEI COMMAND: PLACE

The PLACE command is used to put a Primitive at a position in a Process. The Primitive may be placed in any position within the Process structure except prior to the START symbol or after the END symbol.

COMMAND SYNTAX:

PLACE {Primitive},[position]

P

where:

{Primitive} is a required parameter, indicating any valid Primitive or HOLD.

[position] is an optional parameter indicating any valid position within a Process structure, (i.e., after the START symbol and before the END symbol). The position of a Primitive in a Process is indicated by the numbered column to the left of the flowchart representation of a Process. The default position is immediately prior to the END symbol.

FUNCTION RESULT:

The user is presented with a form that corresponds to the Primitive to be placed.

When the form has been completed, the Process is redrawn if the Primitive is placed on screen and the user is in DRAW mode. The Primitive is placed at the position indicated by the position parameter, if given, and all following Primitives are moved down one position. If the position parameter is omitted, the Primitive is placed immediately prior to the END Primitive. If the Primitive was placed off-screen or the user is in NODRAW mode, the Process is not redrawn, and the user does not see the placement of the Primitive.

6.2.13 PEI COMMAND: REDRAW

The REDRAW command is used to update the current Process display. This command is generally used when the user is in NODRAW mode.

COMMAND SYNTAX:

REDRAW

RED

FUNCTION RESULT:

This command causes the Process display to be redrawn from the location at which the display was last drawn. For example, if the last time the Process displayed was updated at the top of the Process, the display will again be drawn from the top of the Process. The top will be displayed even if the user has made changes to other areas of the Process off screen, as long as those changes were not displayed. Other portions of the Process can be displayed using the TOP, BOTTOM, UP and DOWN commands. The REDRAW command is especially useful when the user is deleting Primitives in NODRAW mode so the user can see what the Process really looks like.

6.2.14 PEI COMMAND: TOP

The TOP command is used to display the first six Primitives in the current Process structure.

COMMAND SYNTAX:

TOP

T

FUNCTION RESULT:

The first six Primitives of the Process structure being edited (or the entire Process if the structure consists of no more than six Primitives) are drawn from the START symbol down. The START symbol is always the first position of a Process structure.

6.2.15 PEI COMMAND: UP

The UP command allows the user to "jump up" the current Process structure an indicated number of positions.

COMMAND SYNTAX:

UP [number of positions]

U

where:

[number of positions] is an optional parameter indicating the number of positions that the structure is to "jump up". If this parameter is not used, the default condition is to "jump up" the Process structure six Primitives, which is analogous to displaying the previous page.

FUNCTION RESULT:

The Process structure jumps up the number of positions indicated by the optional parameter, if given. Otherwise the structure jumps up six Primitives or to the top of the structure if less than six Primitives precede the first position currently displayed.

6.2.16 Terminating a PEI Session

Only one Process can be created or edited during a PEI session. To create or edit other Processes or change to another level the user must terminate the current PEI session and return to the DUI level. This is accomplished by giving the END command described in section 6.2.7. The current working database is left open and control is transferred to the DUI level.

6.3 ARCHITECTURE DESIGN EDITOR (ADE)

The ADE is used to define the layout and interconnection of the physical aspect of a data processing network. It is not necessary to develop an architecture model if the user wishes to model operations without regard to where these operations take place. However, if Items are routed through a system or if Processes at one location trigger Processes in another, then an architecture model is necessary.

The ADE allows the user to create graphically a picture of the system architecture by positioning symbols and connections. It also allows the user to define the legal paths of communication between the connections (and along the connections).

Even if a user has defined a Legal Path Table while creating an architecture, the system offers the option of automatically building a Legal Path Table. The user is queried to resolve any ambiguities. The Legal Path Table is used during the simulation to control the routing of Items that are being passed through the system.

It is important to note that each node and link represented in the architecture is intended to represent some system resource such as a CPU, disk drive, tape drive, or channel. The system automatically creates model Resources for these system Resources. The parameters of such Resources can be altered both in the ADE--though the DEFINE command (see section 6.3.6)--and in the DUI—with the EDIT command (see section 6.1.4).

Hardcopies of a created architecture can be reproduced using a graphics device (see appendix A.4).

6.3.1 Concepts for Using ADE

This section is intended to familiarize the user with the capabilities of the ADE so that he may better understand the description of its use in sections further below.

The view space is divided by vertical and horizontal grids. Grid lines running vertically mark off the position and are numbered starting with zero at the left side. Grids running horizontally mark off the Y position and are identified with numbers, starting with zero at the bottom. Another aid to building the architecture is variable symbol size. The user can specify the size of symbols as he positions them in the view space. The user is provided with commands to change his view screen position, to position nodes which represent system Resources, to delete nodes, and to change symbol names and sizes. A command is provided which allows the user to specify connections between nodes. These connections (or links) are defined as system Resources. Any two nodes may be connected by more than one link, but there may be only one legal path between these two nodes. (Exception: When using Method A, B, or C algorithms to define the Legal Path Table, two node types "TTY" and "LOD" are considered "leaf-nodes" and should have only one connection to one other node. The architecture developed using the ADE becomes the basis

for generating the Legal Path Table which is used to route Items through a system.

The view screen on the HP-2647A terminal, for example, is approximately five inches high by eight and one-half inches wide. This workspace is too small for some systems. The ADE, therefore, gives the user a workspace which is thirteen and two-tenths inches high by 20 inches wide and allows the user to move the viewspace anywhere in this workspace to construct the architecture. The contrast between viewspace and workspace is illustrated in figure 6-6. The workspace is the same size on all terminals supported by AISIM.

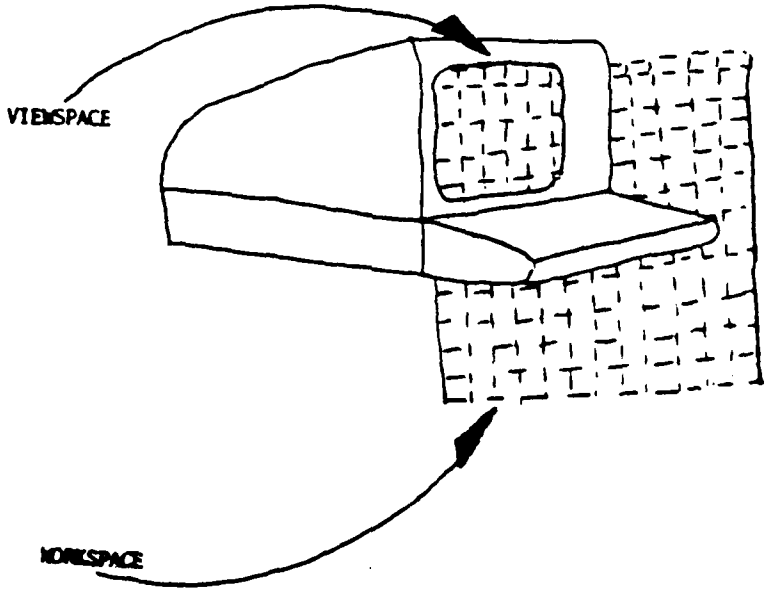


Figure 6-6. Viewspace versus Workspace in ADE

6.3.2 Use of the ADE

The ADE can only be accessed from the DUI level. The ADE level is entered by issuing the following command:

```
ARCH

A
```

Only one architecture is allowed per design database. This prevents the user from specifying an architecture structure that does not relate to the Processes and Resources that have been defined. Experiments using common Processes, Resources, etc. with different architectures can be run by following the procedure listed below:

1) While in the VAX/VMS ready level or AISIM READY level, COPY the project.DBF data file to newproject.DBF data file where: project and newproject are names of PROJECT databases for AISIM models.

2) Enter the ADE to edit the architecture contained in newproject.DBF.

3) Simulations can now be run using the newproject database.

If there is no architecture defined in the design database, the system will provide a blank grid on the screen and a pound sign (#) prompt for the user to enter commands. If an architecture has already been defined, then the old architecture will be displayed and the user will be provided a pound sign (#) prompt for entering commands.

The ADE has DRAW and NODRAW modes which are similar to the PEI DRAW and NODRAW modes. However, if a user is logged on to a VT100 terminal, only NODRAW is available. In NODRAW mode, the user can place and change symbols, connect nodes, and perform all of the functions of the ADE, except that the results of the commands will not be reflected in the screen until the user explicitly redraws the screen with a REDRAW or WINDOW command. If the user is in DRAW mode on a supported terminal, the results of all ADE commands will be reflected in the display. The VT100 is always in NODRAW mode. The default for the other terminals is DRAW mode.

The following pages give a summary of commands available in the ADE and their use. These commands are legal in the ADE level only.

```
CHANGE NAME,{name},{new-name}
       TYPE,{name},{type}
       SIZE,{name},{size}
CHG

CONNECT {node1},{node2},{link}.[F]
CON

DEFINE {symbol-type},[Resource-name]
       PATH,{node1},{node2},{Link1},...,{Linkn}
DEF

DELETE {name1},...,[name-n]
DEL  *

DRAW
DR

END

LIST PATH,{node1},{node2}
     LPT
L

MOVE {node},{x-position},{y-position}
M

NODRAW
N

PLACE {symbol-type},{node},{x-position},{y-position},[size]
P

RECON {link}
R

REDRAW
RED

SAVE

WINDOW {direction1},[n],[direction2],[n]
W
```

Figure 6-7. ADE Command Summary

6.3.3 ADE Symbols

Symbols used to construct an architecture are generic in nature. The shape associated with some symbols is representative of a computer system's hardware elements although no implicit attributes of computer hardware elements are given to the symbols. Attributes defined for a symbol which make it represent an actual physical device must be defined by the user. Attributes are attached to symbols by the DEFINE command.

Symbols in an architecture correspond directly with Resources. This relationship applies to nodes and links. All symbols which are directly connected correspond to an entry in the Legal Path Table.

One other implied relationship applies to the symbols in an architecture. The symbols TTY and LOD are considered to be "terminal" symbols by the Legal Path Table. Therefore, these two symbols have a constraint that they can be connected with only one link to one of the other symbol types. Also, TTY and LOD symbols cannot be directly connected. These constraints are enforced by the LPT generation not the ADE.

The complete symbol set for AISIM architecture is shown in figure 6-8.

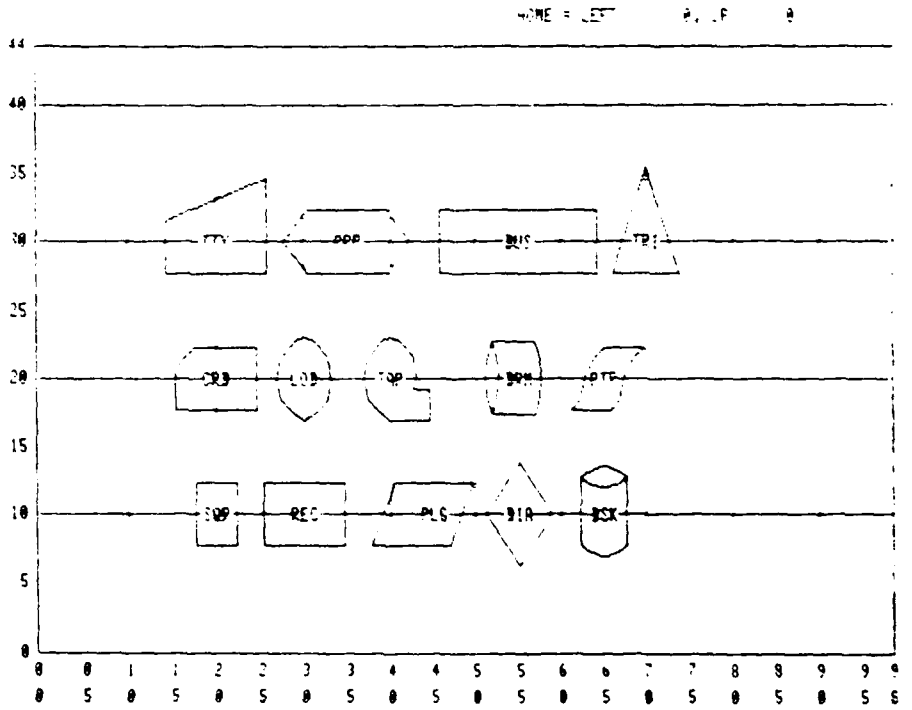


Figure 6-8. Architecture Symbols

6.3.4 ADE COMMAND: CHANGE

The CHANGE command allows the user to modify the name, type, or size of an ADE symbol which represents an architecture node.

COMMAND SYNTAX:

CHANGE NAME,{name},{new-name}

CHANGE TYPE,{name},{type}

CHANGE SIZE,{name},{size}

CHG

where:

{name} is a required parameter indicating the name of the symbol which is to be changed. For the commands CHANGE TYPE and CHANGE SIZE, name must designate a node.

{new-name} is a required parameter specifying a new name for the current named symbol where new name should be 1-8 alphanumeric characters.

{type} is a parameter specifying that the named symbol is to be changed from its current type to "type" which is one of the legal symbol types. The symbol types are shown in figure 6-8.

{size} is a required parameter specifying that the named symbol is to be changed from its current size to "size" where size can be 1-20.

FUNCTION RESULT:

The indicated changes are made to the symbol "name". When the user changes a symbol type or size, there is no impact on the other parameters. When the name is changed, the default size is the number of characters in the name. If the user is in DRAW mode, the symbol is redrawn to reflect the changes.

6.3.5 ADE COMMAND: CONNECT

The CONNECT command is used to show connections between architecture nodes by placing links between them.

COMMAND SYNTAX:

CONNECT {node1},{node2},{link}.[F]

CON

where:

{node1} is a required parameter indicating the first symbol of a from-to pair of symbols to be connected and where node1 is 1 to 8 alphanumeric characters.

{node2} is a required parameter indicating the second symbol of a from-to pair of symbols which are to be connected and where node2 is 1 to 8 alphanumeric characters.

{link} is a required parameter indicating the name of the connection which is to be made and where link is 1 to 8 alphanumeric characters.

[.F] is an optional parameter appended directly to link indicating that the communication link between nodes node1 and node2 is full-duplex. The name of the link must be no longer than eight characters including the ".F" The effect of this is to create two links, a "link.A" and a "link.B". Links defined without this parameter bear a half-duplex default.

FUNCTION RESULT:

If node1 is not in the viewspace when the command is issued, the user will be prompted with the message,

THE FROM NODE MUST BE ON THE SCREEN TO ESTABLISH CONNECT: COMMAND ABORTED:

If node1 is on the viewspace and the user is on a terminal other than a VT100, a cursor (+) is turned on. If the user is on an HP terminal, the cursor appears at node1. If the user is on a TEK4105 terminal, the cursor appears where it was last positioned, or at the lower left corner if it was never moved. At this point, the user has two alternatives:

1) he may cause the system to connect the two symbols with a straight line through their centers by depressing any non-period, alphanumeric character or,

2) he may cause the system to produce a shaped line segment from symbol 1 to symbol 2 by:

a) moving the cursor using the graphics controls, to a position where he wishes to bend the line,

b) typing a period (.),

c) repeating a) and b) until a maximum of five corners have been created.

d) completing the line segment from the last corner to symbol 2 by entering a non-period alphanumeric character.

Alternative 2 allows the user to place symbols randomly and later show connections that would be obscured or confusing if generated by Alternative 1. Connections can be straightened or have corners added to them with the RECON command (see section 6.3.14).

If the user is on a VT100 terminal, the two nodes are automatically connected by a straight line. Bent line connections are not possible.

If the user is in NODRAW mode on a terminal other than a VT100, the connect command operates as stated above for DRAW mode except that the line or line segments are not reflected on the screen. Thus the user can still make connections while in NODRAW mode.

After a connection is defined, two entries are entered in the Legal Path Table. The first is an entry for the path from node1 to node2 via link, and the second entry specifies a path from node2 to node1 via link. If link is defined as full-duplex, then the path from node1 to node2 uses "link.A", while the path from node2 to node1 uses "link.B". (See section on "Define" command). Node1 is then established as the link's from node and node2 is established as the link's to node. All subsequent paths using this full-duplex link will use "link.A" if they go in the direction of the from node to the to node and will will use "link.B" if they go in the opposite direction.

6.3.6 ADE COMMAND: DEFINE

The DEFINE command serves two functions. It is used to define attributes to be associated with symbols (this allows the user to make the logical assignment of physical device characteristics to the Resource). DEFINE is also used to indicate the legal path between nodes in the architecture.

COMMAND SYNTAX:

DEFINE {symbol-type},[Resource-name]

DEFINE PATH,{node1},{node2},{link1},...,{linkn}

DEF PATH

where:

{symbol-type} is the symbol type (sqr,dia,lod,tty,etc.) for which the user wishes to define attributes. Figure 6-8 shows these symbols.

[Resource-name] is an optional parameter that specifies the name of an existing Resource from which the symbol-type attributes are to be copied.

{node1} is the name of the node from which the path is to run.

{node2} is the name of the node to which the path is to run.

{link1},...,{linkn} are the names of the links along which the legal path between node1 and node2 is to run.

FUNCTION RESULT:

If the DEFINE command is issued with the format

DEFINE {symbol-type}

a form will be displayed that shows the parameters currently assigned to this symbol type. The form has the same format as the Resource form in figure 3-6. The user may modify these parameters as desired. After symbol attributes have been defined, any further Resources automatically created in association with the symbol will be given the attributes that were defined for that symbol type.

If the syntax of the command is:

DEFINE {symbol-type},[Resource-name]

the system will present the user with a form to be filled with the attributes of the named Resource. The user can check the data and/or modify it. When entered, the data last displayed in the form will be used to create the attributes of the symbol type.

If the syntax of the command is,

DEFINE PATH {node1},{node2},{link1},...,{linkn}

DEF P

entries in the Legal Path Table will be made. These entries can be inspected with the LIST command (see section 6.3.10).

There are several rules constraining the creation of a legal path in ADE.

First, a point-to-point path is a path between two nodes that are separated from one another by a single link, i.e., there is no other node between them.

Secondly, a sub-path of a given path is any one of the segments of the path that go to the same node as the path but from any one of the nodes the original path passes through. For example, a defined legal path from node1 to node2 to node3 to node4 will have the following sub-paths: (1) from node2 through node3 to node4 and (2) the point-to-point path from node3 to node4. The path from node1 to node3 through node2 is not a sub-path of the original path because it does not go to the same node as the original.

With these two definitions, we can state four quite general rules governing the definition and deletion of legal paths. They are:

1) A Legal Path between two nodes is a collection of Legal Path Table entries of the form:

FROM TO NEXT LINK

which indicates that the path from node FROM to node TO goes to node NEXT via link LINK.

2) There may be only one Legal Path between any two nodes.

3) There must be a path between any two nodes that are directly connected.

4) Use of the two links implied by a full-duplex name for a connection follows these rules:

a) When a connection Con.F is established (actually Con.A and Con.B) with the command,

CONNECT NODE1,NODE2,CON.F

Node1 is established as the from node for that connection and Node2 is established as the to node.

b) Any path which uses the connection CON.F in the direction from its from node to its to node will use CON.A.

c) Any path which uses the link CON.F in the direction from its to node to its from node will use CON.B

d) Establishing the connection between two nodes implicitly defines a point-to-point path between them.

These four rules have a number of restrictions of which the user should be aware:

1) Defining a path from one node to another implies defining paths from all nodes along the path to the last node in the path.

2) Changing a path (redefining, deleting) changes any other paths that use it as a sub-path.

3) A point-to-point path cannot be deleted.

4) When a path between two directly connected Nodes is deleted, a point-to-point path is automatically restored.

5) Deleting a node or link from an architecture removes any paths which use the deleted entity.

6) Changing the name of a node or link changes the name of the entities in the Legal Path Table as well.

7) Cyclic paths are not allowed.

6.3.7 ADE COMMAND: DELETE

The DELETE command allows the user to delete nodes or links in the architecture or parts (or all) of the previously defined Legal Path Table (LPT).

COMMAND SYNTAX:

```
DELETE {name1},...,[name-n]

   PATH {node1},{node2}

   *

DEL
```

where:

{name1} is a required parameter that specifies the node or link to be deleted.

{name-n} is an optional parameter which specifies an additional node or link to be deleted.

{node1} and {node2} are required parameters indicating the nodes between which the legal path is to be deleted.

* indicates the entire architecture is to be deleted.

FUNCTION RESULT:

If the user is in DRAW mode, the following results are seen. When a symbol is being deleted, the symbol and all connections to it are erased from the screen and removed from the database. If a connection is being deleted, the connection is erased from the screen and is removed from the database.

If the user is NODRAW mode, the affected entries are deleted from the database and the screen remains unchanged. When a path between node1 and node2 is deleted from the Legal Path Table, only that path is deleted; any sub-paths which are in this path are unaffected.

6.3.8 ADE COMMAND: DRAW

The DRAW command is used to put the user in the ADE DRAW mode.

COMMAND SYNTAX:

DRAW

DR

FUNCTION RESULT:

The DRAW command sets the ADE mode to DRAW mode. This mode will remain in effect for all future ADE sessions during the current DUI session until changed by a NODRAW command. The DRAW command is not available on a VT100 terminal.

In DRAW mode, all changes made by a user to the architecture which affect the architecture display are immediately reflected in the display. I.e., all nodes and connections are drawn on the screen as they are added to the architecture, and deleted nodes and connections are erased from the architecture.

6.3.9 ADE COMMAND: END

The END command is used to terminate the ADE session.

COMMAND SYNTAX:

END

FUNCTION RESULT:

The END command terminates the edit mode of the ADE session and automatically triggers the generation of a Legal Path Table (LPT). The user will be questioned as to the method of generation for the LPT and for information necessary to clear up ambiguities in its generation before control is returned to the DUI level. The LPT is described in section 6.3.18.

If the user does not wish to generate an LPT, another END command will return control to the DUI level.

6.3.10 ADE COMMAND: LIST

The LIST command enables the user to list the legal paths that have been defined in the architecture.

COMMAND SYNTAX:

```
LIST PATH,{node1},{node2}

    LPT
```

where:

{node1} is the name of the node at which the path to be listed begins.

{node2} is the name of the node at which the path is to end.

FUNCTIONAL RESULT:

If the command syntax is LIST PATH, a format like that below is displayed:

FROM: node3 TO: node2 PATH:

link1,link2,...,linkn

If the command syntax is LIST LPT, the entire Legal Path Table is displayed.

6.3.11 ADE COMMAND: MOVE

The MOVE command allows the user to change the location of a node in the architecture.

COMMAND SYNTAX:

MOVE {node},{x-position},{y-position}

M

where:

{node} is the name of the node to be moved.

{x-position} is the x-coordinate of the new position, i.e., the position to which the node is to be moved.

{y-position} is the y-coordinate of the new position, i.e., the position to which it is to be moved.

FUNCTION RESULT:

If the user is in DRAW mode, the node and all links to or from it will first disappear from the screen. The node will then be redrawn at the new position and the previously defined connections with other nodes will reappear.

If the user is in NODRAW mode, the coordinates of the node will be changed in the database, and the screen will remain unchanged.

6.3.12 ADE COMMAND: NODRAW

The NODRAW command is used to put the user in the ADE NODRAW mode.

COMMAND SYNTAX:

NODRAW

N

FUNCTION RESULT:

The NODRAW command sets the ADE mode to NODRAW mode. This mode will remain in effect for all future ADE sessions during the current DUI session until changed by a DRAW command (section 6.3.8).

In NODRAW mode, no changes which are made to the architecture are reflected in the display until the display is explicitly redrawn by the user. For example, when nodes are placed in the architecture or deleted from the architecture, the changes are made to the database, but the screen remains unchanged. Commands which can be used to update the display are REDRAW (section 6.3.15) and WINDOW (section 6.3.17) commands.

All ADE commands are available while in NODRAW mode, except that on a VT100 terminal, connections can only be straight lines - bent line connections are not allowed. Connections on the VT100 are drawn automatically when the CONNECT (section 6.3.5) and RECON (section 6.3.14) commands are issued.

6.3.13 ADE COMMAND: PLACE

The PLACE command allows the user to position a legal ADE symbol in the view space at specified coordinates.

COMMAND SYNTAX:

PLACE {type},{node},{x-position},{y-position},[size]

P

where:

{type} is a required parameter which specifies one of the legal ADE symbol types. The legal symbol types are shown in figure 6-8.

{node} is a required parameter that indicates the name that is to be displayed and associated with this placement of a symbol and where name is 1 to 8 alphanumeric characters.

{x-position} is a required parameter that specifies the horizontal position of the symbol relative to vertical grid number position 0. The x-position must be within the limits of the view screen.

{y-position} is a required parameter that specifies the vertical position of the symbol relative to horizontal grid position 0. The y-position must be within the limits of the view screen.

[size] is an optional parameter specifying the size of the symbol to be placed. The default size is the number of characters in name. Legal sizes are 1-20.

FUNCTION RESULT:

If the user is in DRAW mode, a symbol of the specified type appears on the view screen at the x, y positions indicated in the command. The symbol name appears within the symbol and the symbol size is regulated by the size parameter.

If the user is in NODRAW mode, the symbol is added to the database, and the screen remains unchanged.

6.3.14 ADE COMMAND: RECON

The RECON command allows the user to alter the shape of a given link, giving it corners, decreasing the number of corners it has, or adding to the number of corners it has.

COMMAND SYNTAX:

```
RECON {link}
R
```

where:

{link} is the name of the link to be redrawn.

FUNCTION RESULT:

If the user is in DRAW mode, the link will disappear, but the cursor (+) will be turned on. The cursor is positioned at the from node on an HP terminal, or at its last location on a TEK4105 terminal (see CONNECT command). As with the CONNECT command (section 6.3.5), the user has two alternatives:

1) cause the system to connect the two symbols with a straight line through their centers by typing any non-period alphanumeric character

2) cause the system to produce a shaped line segment from symbol 1 to symbol 2 by:

   a) moving the cursor using the graphics controls, to a position where he wishes to bend the line,

   b) typing a period (.),

   c) repeating (a) and (b) until a maximum of five corners have been created.

   d) completing the line segment from the last corner to symbol 2 by entering any non-period alphanumeric character.

If the user is on a VT100 terminal, a straight line connection is automatically created between the two nodes and stored in the database, but the screen remains unchanged.

If the user is in NODRAW mode on another terminal, the two options given above are still available. The only difference is that the connection lines are not displayed on the screen as the connection is defined.

6.3.15 ADE COMMAND: REDRAW

The REDRAW command causes the current architecture window to be redrawn to reflect any changes which have been made in NODRAW mode.

COMMAND SYNTAX:

REDRAW

RED

FUNCTION RESULT:

The display is redrawn to reflect the current architecture including all changes made by the user while in NODRAW mode.

6.3.16 ADE COMMAND: SAVE

The SAVE command copies the contents of the working database into the user's permanent database.

COMMAND SYNTAX:

SAVE

FUNCTION RESULT:

The permanent database is replaced with the contents of the working database, and the user is returned to the ADE ready state - # prompt. This command is useful when the user is defining a large system because it allows the user to protect the work done up to the point of issuing the SAVE command.

6.3.17 ADE COMMAND: WINDOW

The WINDOW command allows the user to move the view screen to any position within the legal view space.

COMMAND SYNTAX:

WINDOW {direction1},[n],[direction2],[n]

W

where:

{direction1} is a required parameter that specifies the direction to move the view screen. Legal directions are:

U = up
D = down
L = left
R = right

[direction 2] is an optional parameter that specifies the direction to move the view screen. Legal directions are:

U = up
D = down
L = left
R = right

[n] is an optional parameter that specifies how many grid positions the view screen is to be moved from its present position. If "n" is not given, a default of half the screen width or height is assumed.

FUNCTION RESULT:

After the command has been issued, the screen is cleared, new coordinates are calculated, and the screen is redrawn as seen from the new position. View screen coordinates do not change; only view space coordinates. If the value of "n" is too large causing the view screen to go beyond the limits of the view space, the value of "n" will be truncated to prevent the system from exceeding the view space bounds.

When the ADE is first entered, the view screen is positioned at the upper left corner of the view space.

6.3.18 Termination of an ADE Session

The ADE session is terminated by issuing the command,

END

This completes the edit portion of the ADE session and begins a sequence of events that leads to a return to the DUI Level. Before control is returned to the DUI level, however, the system gives the user the option of creating a new Legal Path Table. The Legal Path Table (LPT) created by the system is based upon the architecture that was created. The LPT consists of a two dimensional array. Entries in the array represent a means of getting from one node to another.

Entries contain two pieces of information:

1) the next node in the path from Node 1 to Node 2
2) the link used to get to the next node.

There are three basic methods of generating a Legal Path Table at the end of an ADE session. In response to the END command, the system questions the user:

BY WHICH METHOD DO YOU WISH TO GENERATE THE LPT (A, B, OR C)?

IF YOU HAVE AN ESTABLISHED LPT OR IF YOU WISH TO SKIP THIS STEP, TYPE "END"

IF YOU DESIRE MORE INFORMATION ON METHODS A, B, OR C, TYPE "INFO"

After the pound sign (#) prompt, the user may enter either "A", "B", "C", "END", "HELP", or "INFO". If the user enters END and a carriage return after this or any subsequent # prompts without responding to the previous prompt question, any currently defined LPT, including none, will remain in effect, and control will return to the DUI level.

If any of the three options is chosen the, previously defined LPT will be deleted from the database and a new LPT will be produced. Since these algorithms may take several minutes, the user is provided with a message that lets him know the system is progressing with the LPT. The prompt initially reads "Generating LPT 1". After so many routes have been found, the message will change to "Generating LPT 2" and so on. The following paragraphs discuss the individual processing performed in response to methods A, B, and C.

METHOD A - Method A directly connects adjacent nodes in the architecture but no other paths are generated. This method is used when message routing paths are not of interest in the model. This method requires the least processing time to generate the LPT. After the user selects method A, the system will begin generation of the LPT. In general, AISIM will not solicit any further information if this method is used.

Method A detects two types of error. If the generator detects an unconnected node, the system will output the following error message:

UNREACHABLE NODE..."node name"

and control is transferred to the DUI level. If multiple links connect nodes, the system will prompt the user for resolution of ambiguous paths. The system will prompt with:

GOING FROM "Node name1" TO "Node name2" CAN GO

1. Through "next Node name" BY CHANNEL "channel name"
2. Through "next Node name" BY CHANNEL "channel name"
   ENTER THE NUMBER OF THE ROUTE YOU WANT TO USE #

All "Through" options will be listed. The choice of path is selected by entering the number of the path after the pound sign (#) prompt. If there are ambiguous paths for other node pairs, the user will be prompted for resolution. If the user should ABORT the LPT generation the following prompt will be displayed:

UNABLE TO SAVE LPT

Control is then passed to the DUI level. If all ambiguities are clarified, the system will complete the generation of the LPT, and issue the following message:

SAVE OF LPT COMPLETE

The user is then at the DUI level.

METHOD B - Method B should be used when there is extensive routing through the architecture. Using Method B, AISIM will algorithmically find all possible legal paths through the system.

This can involve a lot of processing in fully connected architectures because a path from every node to every other node must be defined. For example, if there are 20 nodes then there will be 380 paths, 20 times 19.

The AISIM responses for method B are similar to those described in method A. Because AISIM will fully connect all nodes in the architecture there are bound to be many ambiguous paths. The user will be prompted to resolve all ambiguous paths.

METHOD C - Method C should be used when there is extensive routing through the architecture, also. Using Method C, AISIM will algorithmically find all possible legal paths through the system but will assume that the path for directly connected nodes in the architecture is the direct link. This can substantially decrease the number of paths the user must resolve.

The AISIM responses for method C are similar to those described in method A.

The HELP request causes the system to show the available commands.

The INFO request prints the following:

METHOD A defines as legal paths only connections directly between adjacent Nodes. Longer paths must be handled explicitly in the user Processes.

METHOD B generates all possible paths between each Node pair. You must identify default legal paths for each Node pair.

METHOD C generates all possible paths between each Node except for directly connected Nodes. In the case of adjacent Nodes, the direct connection is assumed as the legal path.

Type END and a carriage return to exit the LPT generation.

In figure 6-9, an AISIM architecture is shown. This architecture connects nine nodes together with 10 links. Using method A, the user is required to resolve 2 ambiguities. Using method B, the user is required to resolve 20 ambiguities. Using method C, the user is required to resolve 12 ambiguities. The Legal Path Tables using each of these methods is shown in figures 6-10 through 6-12.

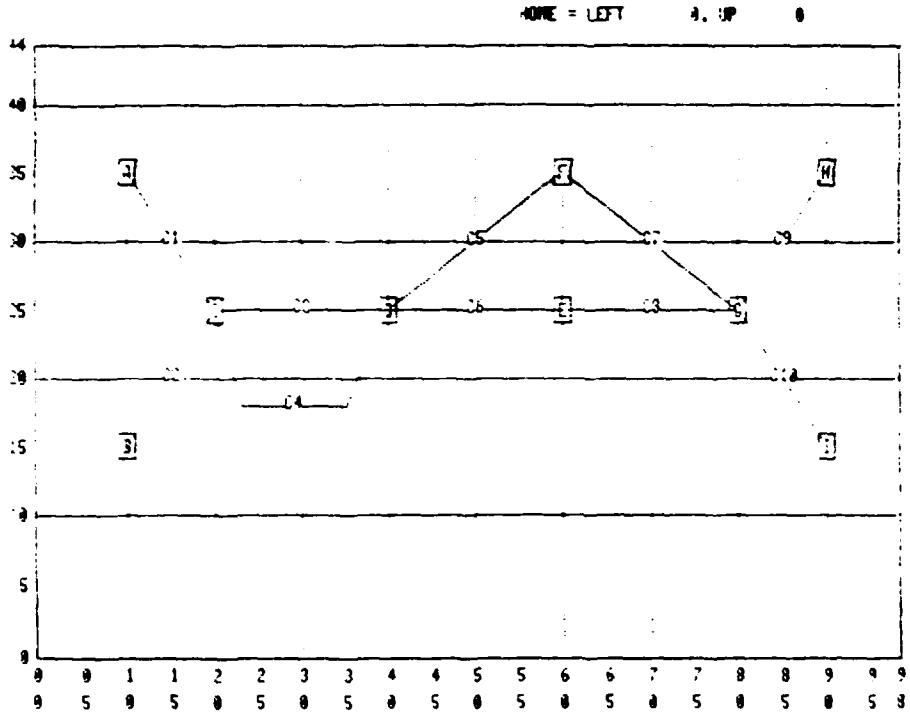
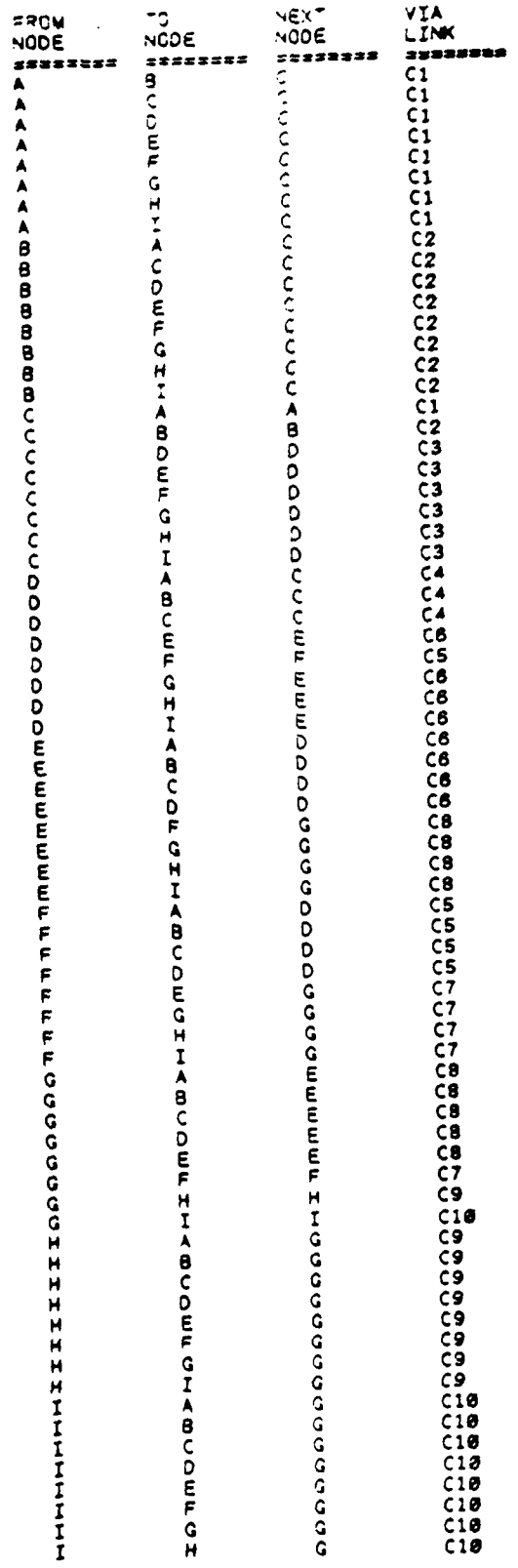


Figure 6-9. Sample Architecture

FROM NODE	TO NODE	NEXT NODE	VIA LINK
A	C	C	C1
B	C	C	C2
C	A	A	C1
C	B	B	C2
C	D	D	C3
D	C	C	C4
D	E	E	C6
D	F	F	C5
E	D	D	C6
E	G	G	C8
F	D	D	C5
F	G	G	C7
G	E	E	C8
G	F	F	C7
G	H	H	C9
G	I	I	C10
H	G	G	C9
I	G	G	C10

Figure 6-10. Sample LPT Generated by Method A



FROM NODE	TO NODE	NEXT NODE	VIA LINK
A	B	B	C1
A	C	C	C1
A	D	D	C1
A	E	E	C1
A	F	F	C1
A	G	G	C1
A	H	H	C1
A	I	I	C1
B	A	C	C2
B	C	C	C2
B	D	D	C2
B	E	E	C2
B	F	F	C2
B	G	G	C2
B	H	H	C2
B	I	I	C2
C	A	A	C1
C	B	B	C2
C	D	D	C3
C	E	E	C3
C	F	F	C3
C	G	G	C3
C	H	H	C3
C	I	I	C3
D	A	E	C4
D	B	E	C4
D	C	E	C4
D	E	E	C6
D	F	F	C5
D	G	G	C6
D	H	H	C6
D	I	I	C6
E	A	D	C6
E	B	D	C6
E	C	D	C6
E	D	D	C6
E	F	F	C6
E	G	G	C6
E	H	H	C6
E	I	I	C6
F	A	D	C8
F	B	E	C8
F	C	F	C8
F	D	G	C8
F	E	H	C5
F	G	H	C5
F	H	I	C5
F	I	I	C5
G	A	D	C7
G	B	E	C7
G	C	F	C7
G	D	G	C7
G	E	H	C8
G	F	I	C8
G	H	I	C8
G	I	I	C8
H	A	G	C8
H	B	H	C7
H	C	I	C9
H	D	H	C10
H	E	I	C9
H	F	I	C9
H	G	I	C9
H	I	I	C9
I	A	G	C9
I	B	H	C9
I	C	I	C9
I	D	H	C10
I	E	I	C10
I	F	I	C10
I	G	I	C10
I	H	I	C10

Figure 6-11. Sample LPT Generated by Method B

FROM NODE	TO NODE	NEXT NODE	VIA LINK
A	B	C	C1
A	C	C	C1
A	D	C	C1
A	E	C	C1
A	F	C	C1
A	G	C	C1
A	H	C	C1
A	I	C	C1
B	A	C	C2
B	C	C	C2
B	D	C	C2
B	E	C	C2
B	F	C	C2
B	G	C	C2
B	H	C	C2
B	I	C	C2
C	A	A	C1
C	B	B	C2
C	D	D	C3
C	E	D	C3
C	F	D	C3
C	G	D	C3
C	H	D	C3
C	I	D	C3
D	A	C	C4
D	B	C	C4
D	C	C	C4
D	E	E	C6
D	F	E	C5
D	G	E	C6
D	H	E	C6
D	I	E	C6
E	A	D	C6
E	B	D	C6
E	C	D	C6
E	D	D	C6
E	F	G	C8
E	G	G	C8
E	H	G	C8
E	I	G	C8
F	A	D	C5
F	B	D	C5
F	C	D	C5
F	E	G	C7
F	G	G	C7
F	H	G	C7
F	I	G	C7
G	A	E	C8
G	B	E	C8
G	C	E	C8
G	D	E	C8
G	E	E	C8
G	F	H	C7
G	H	I	C9
G	I	G	C10
H	A	G	C9
H	B	G	C9
H	C	G	C9
H	D	G	C9
H	E	G	C9
H	F	G	C9
H	G	G	C9
I	A	G	C10
I	B	G	C10
I	C	G	C10
I	D	G	C10
I	E	G	C10
I	F	G	C10
I	G	G	C10
I	H	G	C10

Figure 6-12. Sample LPT Generated by Method C

SECTION 7

ANALYSIS USER INTERFACE (AUI)

After completing a model design using the DUI, the model can be exercised using the commands available in the AUI. During simulation, statistics are kept on Variable values, Item throughput, Resource utilization, queueing delays, Queue lengths, Action times, Process execution, and Process timing. A set of output reports organizes these statistics for printing off line or viewing on-line (while in the AISIM READY level) after completion of the simulation run. Plots of selected model parameters, however, may be drawn on the screen when simulation is halted at a breakpoint, end of period, or end of simulation.

The command issued to enter the AUI from the AISIM READY level contains an optional parameter NOXLATE. If this parameter is omitted, the project database is first translated before a simulation is performed. This translation converts the database into the format required for simulation execution.

If the NOXLATE parameter is used, no translation will take place. The last translation of the project database is used in executing a simulation run. Since another translation is required only if the database was changed (in the DUI) since the last translation, it is not always necessary to repeat the translation process at the start of an analysis session. The NOXLATE option permits skipping of the translation step.

In the translating process, the user is asked the following question, if there is more than one Scenario in the project database:

WHICH SCENARIO DO YOU WISH TO TRANSLATE?

The user must respond with a valid Scenario name, one that has been defined previously in DUI level. A carriage return in response to this question will cause AISIM to list available Scenarios.

If the Scenario name given is invalid the system will respond:

INVALID SCENARIO NAME - REENTER

The user should then enter the correct Scenario name.

When translation of the model and Scenario has completed, the simulator reads the translated database and checks for errors. If the simulator detects one or more errors, the message

ERRORS DETECTED IN MODEL TRANSLATION

is displayed, the AUI level is exited and the user is returned to the AISIM READY level.

At this point the user should enter the EDIT command (described in section 7.4). This automatically invokes the EDT line editor on the project report file. The user should use the Find command of the VAX/VMS EDT editor to list all occurrences of "####". This will result in a list of all errors detected during initialization. Each error documents a problem detected in the model. The EDT line commands used to view the report file are discussed in section 11.3.

If no errors are detected, the following message is displayed:

```
NO ERRORS DETECTED IN MODEL TRANSLATION
YOU MAY NOW ENTER COMMANDS
```

The system provides a # prompt and is ready to accept any of the valid AUI commands. These commands are described in the following pages.

During each of the three phases of analysis - 1) pre-simulation (before the first GO command is issued), 2) mid-simulation (after the first GO command is issued but before simulation termination), and 3) post-simulation (after simulation termination), the user can invoke different commands.

PRE-SIMULATION COMMANDS:

CANBREAK DEFPLOT EDIT END GET GO

INFRES LIST LISTVAL

SAVE (plot definitions) SETBREAK DELETE

MID-SIMULATION COMMANDS:

CANBREAK EDIT END GO LIST LISTVAL

PLOT SAVE SETBREAK

POST-SIMULATION COMMANDS:

END LIST LISTVAL PLOT SAVE

The simulation is started with the GO command.

The SETBREAK and CANBREAK commands are used to establish and cancel stopping conditions (or breakpoints) for the simulation. EDIT is used to make temporary changes to Constants, Variables, and the random number seed values (the keyword is STREAM) upon which stochastic timing and probabilistic branching are based. The Scenario and Loads may be modified by changing the values of parameters specified by Constants. A limited number of Resources in the model sometimes causes a bottleneck which is

evidenced by a waiting line or queue. The effects of this queueing may be eliminated by changing the available Resources to an unlimited quantity. The INFRES command is used to do this on a temporary basis. LIST and LISTVAL are used to display model entities, their attributes, and their values. LISTVAL also allows the user to examine the current random number seeds. The END command returns control to the AISIM READY level.

The DEFPLOT and PLOT commands are used to specify what information is to be gathered for graphs and to request the graph to be displayed at the terminal, respectively. The DEFPLOT command can only be used prior to the start of simulation since the simulator must know what statistics to sample.

Simulation may be performed in periods and is suspended at the end of the number of periods specified. The number of periods to be simulated is specified as an optional period of the GO command. The user is prompted at the end of the period with the message:

```
END OF PERIOD
YOU MAY NOW ENTER COMMANDS
```

and with an audible 'beep' at the terminal.

The user can now make changes in the values of Variables, set breakpoints, display plots, or cancel breakpoints. By suspending the simulation at the end of a period, the user can dynamically interface with the model.

A similar result occurs at a user specified breakpoint, except that the message reads:

```
BREAK POINT REACHED:
(description of the condition of the breakpoint)
YOU MAY NOW ENTER COMMANDS
```

An audible 'beep' is also sounded at this point.

The AUI level commands are described in detail on the following pages.

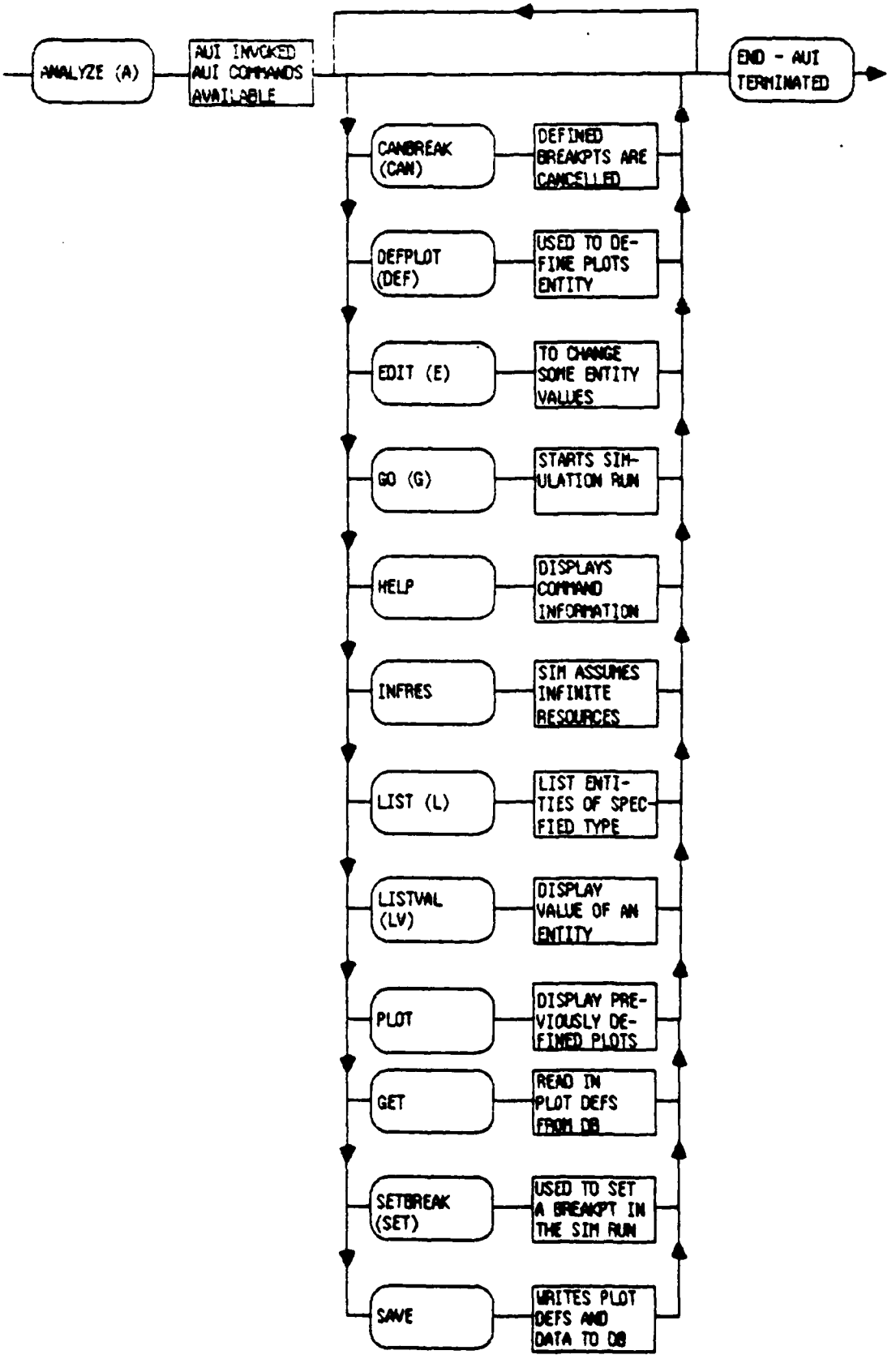


Figure 7-1. Analysis User Interface Commands

```
CANBREAK
CAN

DEFPLOT {entity-type},{entity-name},...,[entity-name]
DEF

DELETE TITLE,{titlenum1}...,[titlenumn]
DELETE TITLE,*
DEL

EDIT {entity-type},{entity-name},{new-value}
E

END

GET DEF,{setname}

GO [n]
G

HELP

INFRES {entity-name},...,[entity-name]
       *

LIST {entity-type/DEF/PLOT/TITLE}
L

LISTVAL {entity-type},{entity-name}
LV

PLOT

SAVE {settype}, {setname},[descr]

SETBREAK {entity-type},{entity-name},{rel-oper},{value}
SET
```

Figure 7-2. AUI Command Summary

7.1 AUI COMMAND: CANBREAK

The CANBREAK command allows the user to cancel a previously defined breakpoint. See the SETBREAK command in section 7.14.

COMMAND SYNTAX:

CANBREAK

CAN

FUNCTION RESULTS:

A previously defined breakpoint is canceled.

7.2 AUI COMMAND: DEFPLOT

DEFPLOT is a pre-simulation command that allows the user to specify what plot data to collect over the period of simulation. The specified plot is added to the present set of plot specifications. This plot data is later graphed with the use of the PLOT command.

COMMAND SYNTAX:

DEFPLOT {entity-type},{entity-name},...,[entity-name]

DEF

where:

{entity-type} is a required parameter indicating a valid entity-type (i.e., Variable, Queue, Resource, Process, Item).

{entity-name} is a required parameter indicating the name of the entity whose value is to be plotted. The user can enter a list of entity names (up to a maximum of eighty characters) of the given entity type at a time. Multiple DEFPLOT commands can be used to define more plots.

FUNCTION RESULT:

This command causes an attribute form to be displayed, from which the user must select one attribute. The list of attributes depends on the entity-type selected.

When the attribute form has been entered, a statistics form is displayed, from which the user must select one statistic. The list of statistics displayed depends on the entity-type and attribute selected.

If only one choice for either an attribute or a statistic exists, the form is not displayed. The forms displayed are shown in figures 7-3 through 7-7. A sample plot is shown in figure 7-8. After the simulation has generated plot data, the plots can be displayed at the user's graphics terminal using the PLOT command (see section 7.12) and printed on a graphics printer (see appendix A.4).

A maximum of ten plots may be specified during any Analysis session.

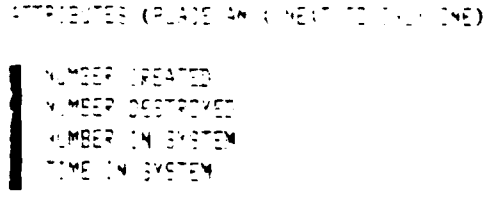


Figure 7-3. DEFPLOT Form for Items

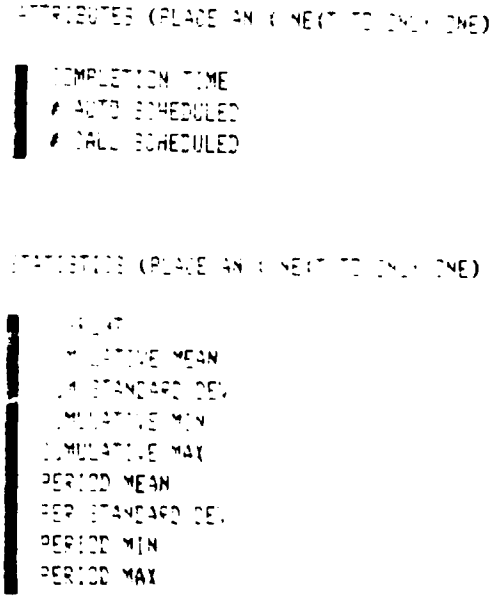


Figure 7-4. DEFPLOT Forms for Process

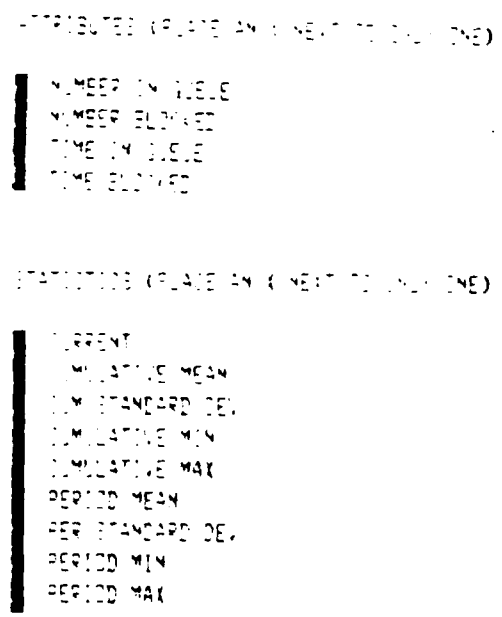


Figure 7-5. DEFPLOT Forms for Queues

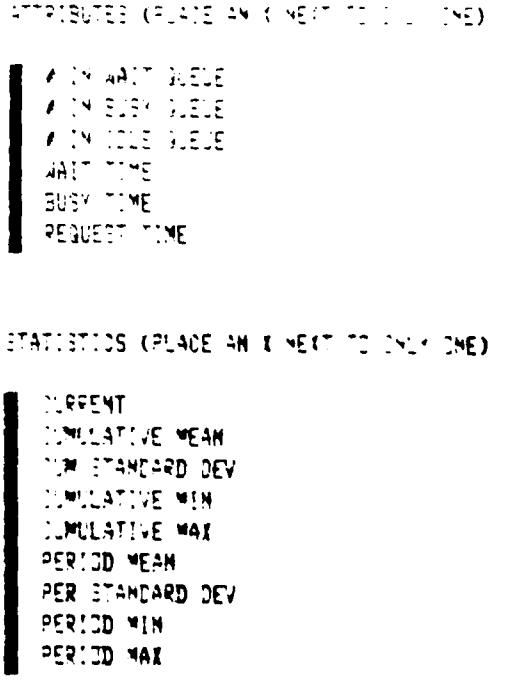


Figure 7-6. DEFPLOT Forms for Resources

STATISTICS (PLACE AN X NEXT TO ITEM ONE)

CURRENT
CUMULATIVE MEAN
CUM STANDARD DEV
CUMULATIVE MIN
CUMULATIVE MAX
PERIOD MEAN
PER STANDARD DEV
PERIOD MIN
PERIOD MAX

Figure 7-7. DEFPLOT Form for Variables

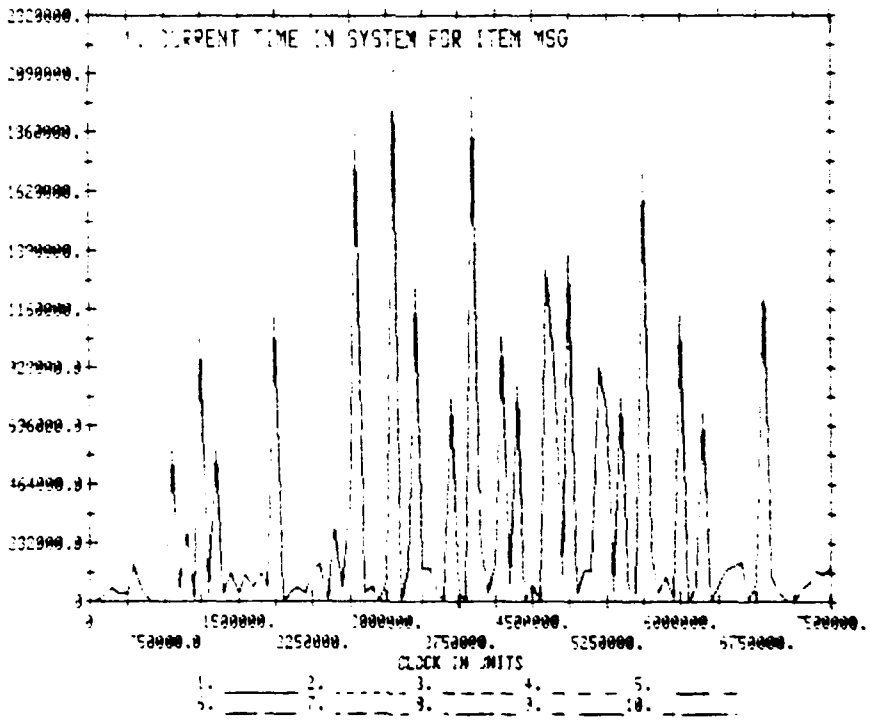


Figure 7-8. Sample Plot

7.3 AUI COMMAND: DELETE

DELETE is a pre-simulation command that allows the user to delete plot definitions which were set up through the DEFPLOT or GET commands.

COMMAND SYNTAX:

```
DELETE  TITLE,{titlenum1},...,[titlenumn]
        TITLE,*

DEL
```

where:

{titlenum1} is a required parameter indicating the number of the plot definition to be deleted. A list of definition numbers may be entered, or entering an asterisk, "*", will cause all of the current plot definitions to be deleted.

FUNCTION RESULT:

This command causes the specified plot definitions to be deleted from those being used for the current simulation run. If the definitions came from a definition set in the database, they still reside in that set in the database, i.e., the database is not modified by this command.

This command allows a user to retrieve plot definitions with the GET DEF command (see section 7.6) and then use only selected definitions for a particular simulation run.

The user can see the numbers of the current plot definitions with the LIST TITLE command (see section 7.10).

7.4 AUI COMMAND: EDIT

The EDIT command in the Analysis mode allows the user to change the value of either a Constant, a Variable, or specification of the random number stream used to represent probabilistic events. The value of a Variable with an alpha literal as its initial value cannot be changed with this command.

COMMAND SYNTAX:

EDIT {entity-type},{entity-name},{new-value}

E

where:

{entity-type} is a required parameter indicating which type of entity is to be changed (either Constant, Variable, or Stream).

{entity-name} is a required parameter indicating the name of the Constant, Variable or Stream (Branch, Load, or Action) which is to be changed.

{new-value} is a required parameter indicating the new value of a Constant or Variable or for STREAM, the new random number stream. The new value may be expressed in one to twelve digits, and includes the value zero. The legal values of "new value" when specifying a random number stream are 1 through 10.

NOTE: Constants may be changed only before the start of the first simulation period. Variables and Streams may be changed before the start of any simulation period or at a breakpoint.

FUNCTION RESULT:

The value of the Constant or Variable or Stream is changed to the new value, and remains at that value until changed by another EDIT command. This command only affects the current translation of the database; therefore, at the end of an Analysis session the Constant or Variable or Stream is restored to its original value.

If the value of the Stream is not changed, default values are:

Action: 3, for random Action durations

Branch: 2, for the PROB Primitive

Load: 1, for random intervals between a Loads' triggering of another Process instance.

7.5 AUI COMMAND: END

The END command is used to terminate an Analysis session.

COMMAND SYNTAX:

END

FUNCTION RESULT:

This command causes all displays to be cleared and, if plots were generated, asks the user "Do you wish to save plot definitions? (Y/N)" and "Do you wish to save plot data? (Y/N)" If the answer to a question is yes, the user is prompted for the required information before control returns to the AISIM READY level. Plot data and definitions are stored in a file called project.PLT where project is the name of the user's project database. Upon termination of an Anaylsis session, a copy of the output report is automatically printed.

7.6 AUI COMMAND: GET

The GET command allows the user to retrieve a previously saved set of plot definitions and add them to the current plot specification. The plot specification defines what plot data will be collected during the simulation. The LIST DEF command may be used to obtain a list of the available plot definition sets.

COMMAND SYNTAX:

```
GET DEF,{setname}
```

where:

{setname} is the name of the set containing the plot definitions. The GET command may be issued only before the first simulation period.

FUNCTION RESULT:

The set of plot definitions is retrieved and made a part of the current set to be used by the Analysis function.

The LIST DEF command (see section 7.10) may be used to obtain a list of the available plot definition sets.

7.7 AUI COMMAND: GO

The GO command allows the user to start or resume a simulation run.

COMMAND SYNTAX:

GO [n]

G

where:

[n] is an optional parameter that specifies how many periods the simulation is to run. If not given, the default result is that the entire simulation defined by the selected Scenario is executed. If an n greater than the number of periods specified in the Scenario is entered, the simulation executes all periods specified in the Scenario and no more.

FUNCTION RESULT:

This command, which is valid before any simulation period or at a breakpoint, begins or resumes the simulation of the translated Scenario.

If used to resume the simulation, resumption occurs at the breakpoint or at the beginning of the next simulation period.

7.8 AUI COMMAND: HELP

The HELP command lists, on the user's terminal, the commands that are valid during each of the three different stages of an Analysis session (prior, during, or after simulation).

COMMAND SYNTAX:

HELP

FUNCTION RESULT:

The HELP command may be invoked prior to, during, or after a simulation run. When invoked, only those commands that are valid at that point in the Analysis session are displayed.

This command is valid at any time during an Analysis session.

During each of the three phases of analysis, the user receives a different output from the HELP command.

HELP invoked prior to sim:

GO G END LIST L LISTVAL LV

EDIT E SETBREAK SET CANBREAK CAN DEFPLOT DEF

INFRES GET DELETE DEL

HELP invoked during sim:

GO G END LIST L LISTVAL LV

EDIT E SETBREAK SET CANBREAK CAN PLOT

SAVE S

HELP invoked after sim:

END LIST L LISTVAL LV PLOT

SAVE S

7.9 AUI COMMAND: INFRES

The INFRES command causes the simulation to assume the existence of infinite available Resources for specified Resources.

COMMAND SYNTAX:

```
INFRES   {entity-name},...,[entity-name]
          *
```

where:

{entity-name} is a required parameter indicating the name of the Resource for which unlimited units are available. A list of up to eight Resource names at a time may be entered, or an asterisk, "*", can be used to indicate infinite resources for all Resource entities in the model. The INFRES command can be entered multiple times to set infinite resources for more entities.

FUNCTION RESULT:

This command, which is only valid before the start of the first simulation period, allows the assumption that infinite Resources are available for the specified Resources during the Scenario being simulated.

7.10 AUI COMMAND: LIST

The LIST command displays all entities of a specified type. Included with each entity is its name and its description.

COMMAND SYNTAX:

LIST {entity-type}

L

where:

{entity-type} is a required parameter indicating any of the specific model entities listed below.

ENTITY	ABBREVIATION
CONSTANT	C
RESOURCE	R
PROCESS	P
VARIABLE	V
ITEM	I
QUEUE	Q
PLOT	none
DEF	none
TITLE	none

This command is valid at any time during an Analysis session.

FUNCTION RESULT:

The user is presented with a list of all existing entities of the requested type. If the argument is PLOT, a list of saved plot sets is given. If the argument is DEF, a list of the saved plot definition sets is given. If the argument is TITLE, a list of the plots defined for the current simulation is given.

7.11 AUI COMMAND: LISTVAL

The LISTVAL command allows the user to display the current statistics for the named entity.

COMMAND SYNTAX:

LISTVAL {entity-type},{entity-name}

LV

where:

{entity-type} is a required parameter indicating a valid Analysis system entity-type. The valid entity types are the following:

Clock	
Constant	C
Item	I
Process	P
Queue	Q
Resource	R
Stream	S
Variable	V

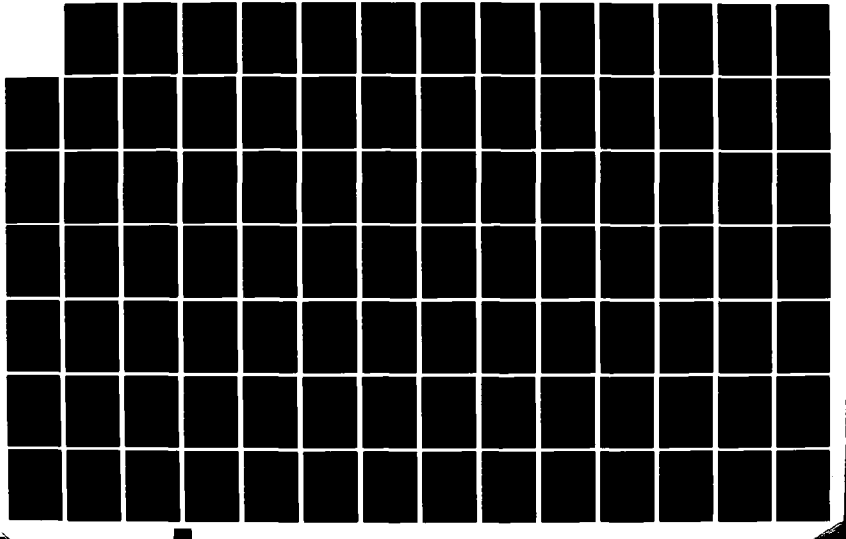
{entity-name} is a required parameter indicating the name of the entity whose value is to be listed. When requesting the Stream for Loads, Branches or Actions, or the Clock, this field is omitted.

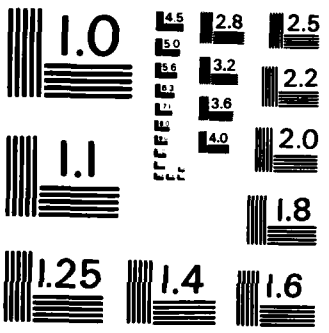
FUNCTION RESULT:

The name of the entity requested is printed out with a listing of all statistics for that entity.

The prompt "**** Enter YES/Y to continue, NO/N to abort ****" is displayed. If the user wishes to end the LISTVAL listing, "NO" is entered and the AUI READY prompt is displayed. If "YES" is entered the next page of the listing is displayed, if there is one, with the prompt displayed again. If there is no further data to be displayed, the user is returned to the AUI ready level.

AD-A161 556 ASIM (AUTOMATED INTERACTIVE SIMULATION MODELING SYSTEM) 3/4
VAX VERSION USER'.. (U) HUGHES AIRCRAFT CO FULLERTON CA
GROUND SYSTEMS GROUP S KNEEBURG FEB 85 ESD-TR-85-127
UNCLASSIFIED F33615-81-C-5098 F/G 9/2 NL





7.12 AUI COMMAND: PLOT

The PLOT command allows the user to produce a graph of the plot data collected during the simulation.

COMMAND SYNTAX:

PLOT

FUNCTION RESULT:

This command, which is valid at the end of a simulation period or at a breakpoint, causes the display of a form containing the plot titles which were defined using the DEFPLOT command (see section 7.2). From this form the user may select any, all, or none of the listed titles.

When the selected titles have been entered, the user is presented with the plot grid. The selected plots are produced and the user is prompted for more Analysis mode commands.

Each of the plots is produced in a unique line pattern.

Once displayed on the terminal, graphs can be transferred to a hardcopy device (see appendix A.4). An example of the form that is displayed to allow the user to select a plot is shown in figure 7-9. A sample plot is shown in figure 7-10.

NOTE: Due to limits imposed by graphics screen resolution, only a sample of the data points produced by the simulation are included in the plot (see appendix A.3).

PLACE AN X NEXT TO THE TITLES YOU WISH TO PLOT

X CURRENT TIME IN SYSTEM FOR ITEM MSG
X CURRENT # IN WAIT QUEUE FOR RESOURCE CH
X CUMULATIVE MEAN # IN WAIT QUEUE FOR RESOURCE CH
X CURRENT NUMBER IN SYSTEM FOR ITEM MSG

Figure 7-9. Sample Form for Selecting Plots

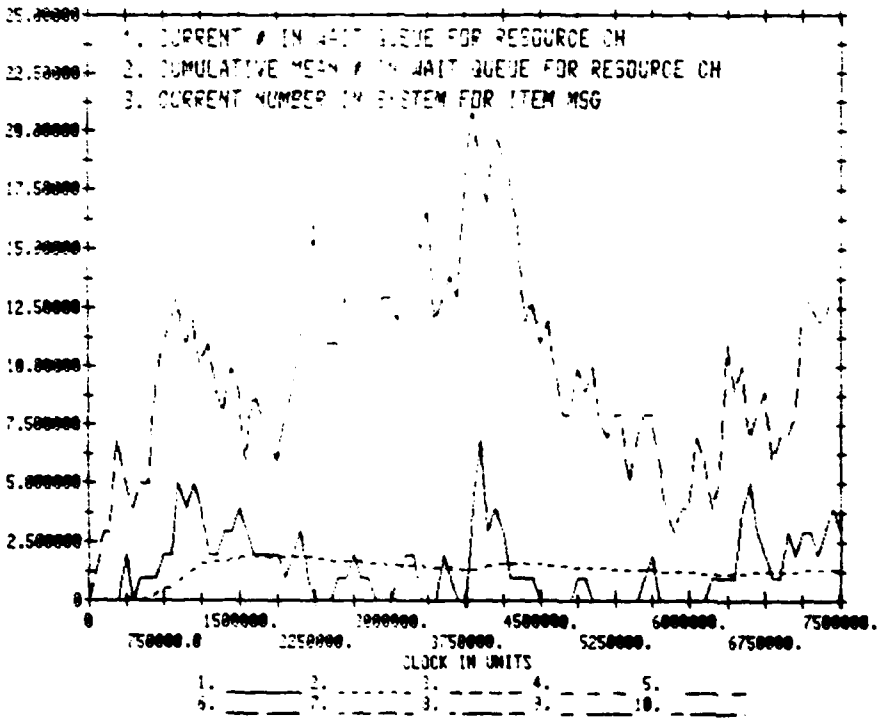


Figure 7-10. Sample Plot

7.13 AUI COMMAND: SAVE

SAVE is used to save current plot definitions or plot data and transfer them to the Analysis database.

COMMAND SYNTAX:

SAVE {settype},{setname},[descr]

where:

{settype} is

1. DEF to save plot definitions, or
2. PLOT to save plot data.

{setname} 1 to 8 character name to be given to the set.

[descr] is a description of the set.

FUNCTION RESULT:

Plot definitions or plot data are flagged to be saved in the Analysis database when the Analysis session is terminated. If {setname} already exists, the user is queried to reuse the old set. A "yes" response will replace the old set with the new set. A "no" response will cause a prompt for a new set name.

7.14 AUI COMMAND: SETBREAK

The SETBREAK command allows the user to set a single breakpoint in the simulation run that is executed when a defined relationship has been satisfied.

COMMAND SYNTAX:

SETBREAK {entity-type},{entity-name},{rel-oper},{value}

SET

where:

{entity-type} is a required parameter indicating which type of entity is to be tested (Variable, Resource, or Process).

{entity-name} is a required parameter indicating the name of the entity to be tested.

{rel-oper} is a required parameter indicating the relational operator (EQ, NE, LE, GT, GE, LT) of the test.

{value} is a required parameter used to set the value for which the named entity is to be tested. This value may be expressed in one to twelve digits, and includes the value zero.

FUNCTION RESULT:

A breakpoint is usually used in verification of a model or to examine Variable values. Typically, a simulation run executes start to finish and does not allow the user to examine the simulation state at specific times during simulation. The breakpoint allows the user to halt the simulation and examine its state based upon the value of some system element.

This command causes an attribute form to be displayed, from which the user must select one attribute. The list of attributes depends on the entity-type selected.

When the attribute form has been entered, a statistics form is displayed, from which the user must select one statistic.

If there is only one choice for either an attribute or a statistic, the form is not displayed. Attribute and statistic forms are shown in figures 7-4, 7-6, and 7-7.

This command is valid at the beginning of a simulation period or at a breakpoint.

When a breakpoint is reached, it is automatically cleared.

7.15 TERMINATION OF AN AUI SESSION

An AUI session is terminated and control is transferred to the AISIM READY level through the command:

END

FUNCTION RESULT:

When the END command is issued, any plot data or plot definitions which the user saved during the AUI session are placed in the user's Analysis database. Any attempts to reuse plot data or plot definition sets are resolved at this time. The user is then returned to the AISIM READY level.

SECTION 8

REPLOT USER INTERFACE (RUI)

The Replot User Interface (RUI) allows the user to:

(1) plot data saved from previous analysis runs,

(2) to delete old plot data and plot definition sets from the data base.

(3) create new plot data sets from data previously saved in separate plot data sets.

When plot data is retrieved from the Analysis database via the GET command (see section 8.4), the plot data is stored in a temporary plotset. This temporary plotset exists for the current Replot session only. Data from different Analysis runs may be retrieved from the database. All of the data is then stored together in the temporary plotset, and may be plotted on the same graph. The SAVE command (see section 8.7) will store all of the data in the temporary plotset into a new, permanent plotset in the analysis database. The temporary plotset can be cleared out (i.e., plot data in it is deleted) using the CLEAR command (see section 8.1). The CLEAR command does not affect permanent data stored in the database.

Once displayed on the terminal, plots can be transferred to a hardcopy device (see appendix A.4).

The RUI level commands are described in detail on the following pages.

```
CLEAR

DELETE {settype},{setname}
DEL

END

GET PLOT,{plotset}

LIST {entity-type}
L

PLOT

SAVE PLOT,{plotset},[description]
S
```

Figure 8-1. RUI Command Summary

8.1 RUI COMMAND: CLEAR

CLEAR is used to delete plot data in the temporary plotset and to clear the screen.

COMMAND SYNTAX:

CLEAR

FUNCTION RESULT:

The temporary plotset is emptied, and the screen is cleared. Plots saved in the database are unaffected.

8.2 RUI COMMAND: DELETE

DELETE is used to delete a set of plot definitions or plot data from the Analysis data base.

COMMAND SYNTAX:

DELETE {settype},{setname}

DEL

where:

{settype} is:

DEF to delete plot definitions, or

PLOT to delete plot data.

{setname} is the name of the set to be deleted.

FUNCTION RESULTS:

The specified set of plot data or plot definitions are deleted from the Analysis data base. The current temporary plot set is unaffected.

8.3 RUI COMMAND: END

END is used to exit the RUI.

COMMAND SYNTAX:

END

FUNCTION RESULT:

The user is returned to the AISIM READY level.

8.4 RUI COMMAND: GET

GET is used to retrieve a set of plot data and to make it part of the current set of plots to be displayed by the PLOT command.

COMMAND SYNTAX:

GET plot,{plotset}

where:

{plotset} is the name of the set containing the desired plot data.

FUNCTION RESULT:

The set of available plots is displayed. The user is then prompted for the plot(s) to be retrieved for use by the PLOT command.

The names of the plot data sets may be listed using the LIST command (see section 8.5).

8.5 RUI COMMAND: LIST

LIST is used to list all entities of the specified type.

COMMAND SYNTAX:

LIST {entity-type}

where:

{entity-type} is a required parameter indicating a valid entity type. It can be one of the following:

DEF to list plot definition sets

PLOT to list plot data sets

TITLE to list current plot titles

FUNCTION RESULTS:

Names of all entities of the requested type are displayed.

8.6 RUI COMMAND: PLOT

The PLOT command is used to display a plot of the activity of an entity.

COMMAND SYNTAX:

PLOT

FUNCTION RESULT:

The set of available plots is displayed. The user is then prompted for the plot(s) to be graphed.

When the selected plot titles have been entered, the appropriate plot is displayed. Each of the plots is produced in a unique line pattern. If only one plot is defined, it will be displayed with no prompting.

Once displayed on the terminal, plots can be transferred to a hardcopy device (see appendix A.4).

8.7 RUI COMMAND: SAVE

The SAVE command is used to save the data in the current temporary plot set into a permanent plotset in the database.

COMMAND SYNTAX:

```
SAVE {setname},[descr]
S
```

where:

{setname} is a 1 to 8 character name to be given to the set

[descr] is a description of the set

FUNCTION RESULT:

The plot data contained in the temporary plot set (as a result of previous GET PLOT commands) is saved into the new plot set. This command enables the user to combine plots from various simulation runs into a single plotset.

SECTION 9

HARDCOPY USER INTERFACE (HUI)

The Hardcopy User Interface (HUI) is used to plot the flowcharts for one, several, or all Processes in the specified project database. In order for the Hardcopy Function to be exercised, the following conditions must be in effect:

For an HP2647 terminal:

1. An HP2631G Graphics Printer must be connected to the HP2647A Graphics Terminal with the HP-IB communications bus.

2. The HP-IB bus address of the printer must be set to one (1).

3. The printer must be turned on and set to ON LINE mode.

4. For proper formatting, the length of the paper in the printer must be either 8 1/2 inches or 11 inches long.

For a TEK4105 terminal:

1. A TEK4695 graphics copier must be connected to the TEK4105 terminal.

For an HP2623 terminal:

1. The internal printer must be functional.

The HUI is entered from the AISIM READY level by typing the command:

HCOPY [PROJECT(project)] [TERM(terminal)]

where:

[PROJECT(project)] is an optional parameter indicating the project database with the Processes of interest. If omitted, the project is assumed to be the last project specified in a previous AISIM READY level command.

[TERM(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified. The valid terminal types are the following:

HP - HP2647A terminal
HP23 - HP2623 terminal
TEK - TEK4105 terminal

The first information that the HUI requests is:

```
PLOT ALL THE PROCESSES IN DATABASE? (YES OR NO)
```

The user responds with "NO" to specify selected Processes for plotting. A "YES" response will cause the system to automatically plot all of the Processes contained in the project data base.

The system then requests information about the printing medium for an HP2647A terminal:

```
ENTER PRINTER PAGE SIZE (A/B):
A) 8 1/2 INCHES
B) 11 INCHES.
LENGTH=
```

Depending on the paper in the graphics printer, the user responds by entering "A" or "B". This information is used by the HUI to center the Process graphics on the page and to insure correct form feeding. Entering any other option besides "A" or "B" causes the prompt to be reissued.

The user is then instructed to:

```
POSITION THE PAPER PERFORATION ALONG THE T.O.F. INDICATOR
LINE ON THE PRINTER AND DEPRESS THE CARRIAGE RETURN.
```

By doing this, the user sets up the proper alignment of the paper in the printer and initiates execution of the Hardcopy plotting software.

When the carriage return has been entered, the HUI begins the plotting procedure by initializing the HP2631G printer with the correct form information. This initialization is usually characterized by a rapid movement of the print head.

If the user is on an HP2623 terminal, only the following prompt occurs to start the Hardcopy operation:

```
POSITION THE PAPER PERFORATION ALONG THE T.O.F. INDICATOR
LINE ON THE PRINTER AND DEPRESS THE CARRIAGE RETURN.
```

When the user presses the carriage return, AISIM initiates execution of the Hardcopy plotting software.

Note: the following terminal configuration must be set up on the HP2623 terminal in order to run the Hardcopy program. These settings need to be set only once unless their configuration is changed at some later time. First press the following function keys:

```
<AIDES>
<CONFIG KEY>
<DATACOMM CONFIG>
```

Then tab through the form display to the

RECV PACE

field. Press the NEXT key until the field reads

XON/XOFF

Then tab to the

XMIT PACE

field. Press the following two keys:

<CONFIG KEY>
<TERMINAL CONFIG>

Tab through the form to the

INHNDSHK(G)

key and press the NEXT key until the field reads

YES

The terminal is now set up for the Hardcopy function.

If the user is on a TEK4105 terminal, the following information is requested:

ENTER PRINTER PAGE SIZE (A/B):
A) 8 1/2 INCHES
B) SMALLER COPY SIZE
LENGTH=

This information is used to create standard size flow diagrams or reduced size diagrams.

The user is then instructed to:

POSITION THE PAPER PERFORATION ALONG THE T.O.F. INDICATOR LINE ON THE PRINTER AND DEPRESS THE CARRIAGE RETURN.

When the user presses the carriage return, AISIM initiates execution of the Hardcopy plotting software.

If the user has requested automatic plotting of all of the Processes, they are plotted in alphabetic order.

If the user asked to select Processes, the following prompt is given:

PROCESS NAMES TO PLOT: (CR TO EXIT)

The user then supplies the name of the Process he wishes plotted, followed by a carriage return. The Process is plotted and the HUI responds with:

<Process name> PLOTTED

The system will then give the selection prompt again for another Process to be plotted. The user continues entering Process names one at a time, followed by a carriage return, or exits the HUI by entering a carriage return only.

The way in which the HUI plots a Process in either of the two modes described above is as follows:

1. The first screen of Primitives in the Process are painted on the screen of the terminal.

2. The Process name is written at the top of the page.

3. If the first page of the Process is being plotted, the Process description is also written across the top of the page.

4. The Process graphics are transferred from the terminal screen to the page in the printer and a form feed is generated.

5. If there are no more Primitives in the Process, the plotting is terminated for the Process; otherwise, the terminal screen is erased, and next six primitives are painted on the screen, and steps 2 through 5 are repeated.

When the HUI has finished plotting all of the requested Processes, the message "ALL DONE" is printed and the user is returned to the AISIM READY level.

SECTION 10

LIBRARY USER INTERFACE (LUI)

The Library User Interface allows the user to do the following:

1. Move entities from a project database into a storage area called a "buffer" using the MERGEOUT sublevel of the LUI.

2. Move entities from a buffer into the database of another project using the MERGEIN sublevel of the LUI.

3. Move entities from a buffer into a library of entities using the CHECKIN sublevel of the LUI.

4. Move entities from a library to a buffer using the CHECKOUT sublevel of the LUI.

5. Convert a pre-version 4.0 project database to a version 4.0 compatible project database.

Two libraries are available. One is a user library in which a user can place entities for private use. Another is an AISIM system library which contains models available for public use. Models are groups of AISIM entities which represent some function or group of functions (see the message routing submodel, appendix D). There are restrictions on the placement of entities in the system library because it is desirable to insure that the public models are not lost or tampered with. For this reason, general users cannot modify the AISIM system library. Access is restricted to the AISIM administrator.

The LUI sublevel is accessible from the AISIM READY level by issuing the command:

```
LIBRARY
```

The system will then respond with the prompt:

```
LIBRARY READY
```

and the user may invoke any of the five LUI sublevels listed in the LUI Command Summary figure 10-1. Figure 10-2 shows the actions of the various LUI functions.

CHECKIN CI	[BUFFER(buffer)] [B(buffer)]	[LIBRARY(library)] [L(library)]	[TERM(terminal)] [T(terminal)]
CHECKOUT CO	[BUFFER(buffer)] [B(buffer)]	[LIBRARY(library)] [L(library)]	[TERM(terminal)] [T(terminal)]
CONVERT CON	[PROJECT(project)] [P(project)]	[TERM(terminal)] [T(terminal)]	
MERGEIN MI	[PROJECT(project)] [P(project)]	[BUFFER(buffer)] [B(buffer)]	[TERM(terminal)] [T(terminal)]
MERGEOUT MO	[PROJECT(project)] [P(project)]	[BUFFER(buffer)] [B(buffer)]	[TERM(terminal)] [T(terminal)]

Figure 10-1. LUI Command Summary

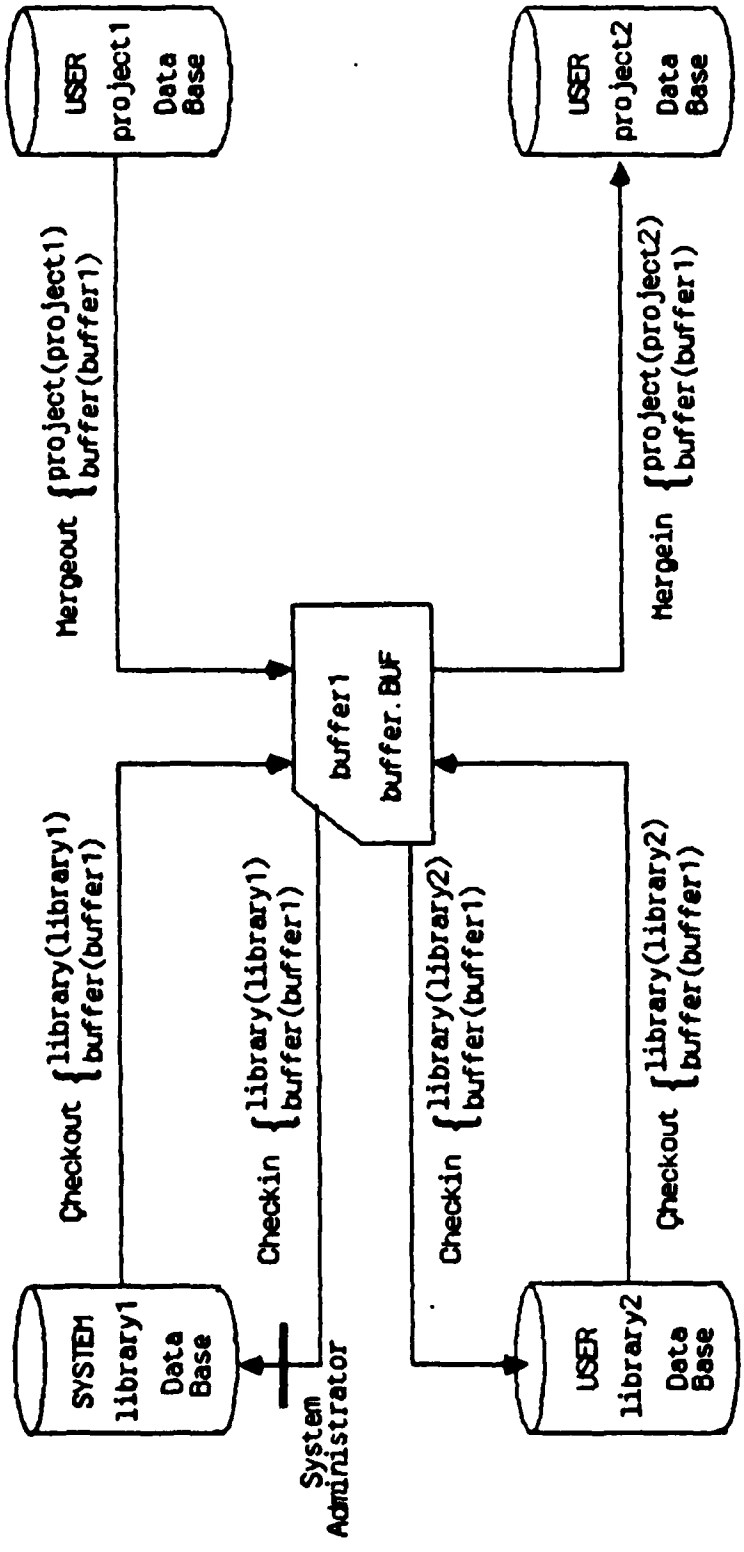


Figure 10-2. Library Utility Data Flow Diagram

10.1 LUI COMMAND: CHECKIN

To move the contents of the buffer to a library for permanent storage, one issues the CHECKIN command. The user is prompted for the name of the model to be checked in, as well as an optional reference number and description.

To enter the CHECKIN sublevel, issue the command:

```
CHECKIN [BUFFER(buffer)] [LIBRARY(library)] [TERM(terminal)]

CI      [B(buffer)]      [L(library)]       [T(terminal)]
```

where:

[B(buffer)] is an optional parameter indicating the buffer from which entities are to be taken. If omitted, the buffer is assumed to be the last buffer specified in a previous LIBRARY READY level command.

[L(library)] is a required parameter indicating the library into which the entities are to be entered.

[T(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

```
HP   - HP2647A or HP2648a terminal
HP23 - HP2623 terminal
TEK  - TEK4105 terminal
VT   - VT100 terminal
```

FUNCTION RESULT:

The system queries the user for a required model name and an optional document reference number and description. After getting this information, the entities in the buffer are put into the library specified under the given model name.

10.2 LUI COMMAND: CHECKOUT

To copy a model stored in a library to a buffer, one enters the CHECKOUT command. At this point the user can obtain a list of the models contained in the library or a list of the given entity types contained in a named model through the LIST command (see section 10.2.5). Models are copied individually through the EXTRACT command which specifies the model to be copied. A HELP command is available. The CHECKOUT sublevel commands are described in detail in sections 10.2.1 through 10.2.5.

To enter the CHECKOUT sublevel, issue the command:

```
CHECKOUT [BUFFER(buffer)] [LIBRARY(library)] [TERM(terminal)]

CO       [B(buffer)]      [L(library)]       [T(terminal)]
```

where:

[B(buffer)] is an optional parameter naming the buffer into which entities are to be placed. If omitted, the buffer is assumed to be the last buffer specified in a previous LIBRARY READY command.

[L(library)] is a required parameter indicating the library from which the entities are to be taken.

[T(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

```
HP   - HP2647A or HP2648A terminal
HP23 - HP2623 terminal
TEK  - TEK4105 terminal
VT   - VT100 terminal
```

FUNCTION RESULT:

The model or models specified by the user are written out to the buffer. From the buffer they can be included in a project with the MERGEIN command.

```
DELETE {model-name}
D

END
E

EXTRACT {model-name}
EXT

HELP

LIST {model-name}
     *
```

Figure 10-3. Checkout Command Summary

10.2.1 CO COMMAND: DELETE

The DELETE command instructs the system to delete a specified model from a user's library.

COMMAND SYNTAX:

DELETE {model-name}

D

where:

{model-name} is the name of the model to be deleted from the library.

FUNCTION RESULT:

The specified model is deleted from the library. If a user attempts to delete a model from the system library, the following message is displayed: "THIS ACCOUNT IS NOT AUTHORIZED TO MODIFY THE SYSTEM LIBRARY."

10.2.2 CO COMMAND: END

The END command causes the system to exit the CHECKOUT sublevel and return the user to the LIBRARY READY Level.

COMMAND SYNTAX:

END

E

FUNCTION RESULT:

If any models were selected for extraction, the entities are written to a buffer.

The system then returns to the LIBRARY READY level.

10.2.3 CO COMMAND: EXTRACT

The EXTRACT command instructs the system to copy a model from a library into a buffer.

COMMAND SYNTAX:

EXTRACT {model-name}

EXT

where:

{model-name} is the name of the model to be placed in the buffer.

FUNCTION RESULT:

The model specified is copied from the current library into the current buffer.

10.2.4 CO COMMAND: HELP

The HELP command enables the user to obtain a menu of the other commands available in the CHECKOUT sublevel.

COMMAND SYNTAX:

```
HELP
```

FUNCTION RESULT:

A menu of available commands is printed on the screen.

10.2.5 CO COMMAND: LIST

The LIST command enables the user to obtain a list of the models contained in a system or user library, or to list the entities in a particular model.

COMMAND SYNTAX:

LIST {model-name}

*

␣

where:

{model-name} is the name of a model in the library

* is a literal parameter, indicating all models in the library.

FUNCTION RESULT:

If the parameter {modelname} is used, the system will display a list of the names of the entities in the indicated model. After the names of each entity type are displayed, the user is given the option of continuing to list model entities or of returning to the CHECKOUT ready level. If the parameter * is used, the system will display a list of the names of all the models in the library.

10.3 LUI COMMAND: CONVERT

The CONVERT command enables a user to convert a pre-version 4.0 project database into a 4.0-compatible database. Old databases are incompatible with version 4.0, so all old databases must be converted before they can be used with version 4.0.

COMMAND SYNTAX:

```
CONVERT [PROJECT(project)] [TERM(terminal)]
CON     [P(project)]       [T(terminal)]
```

where:

[P(project)] is a required parameter indicating the name of the project being converted.

[T(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

```
HP   - HP2647A or HP2648A terminal
HP23 - HP2623 terminal
TEK  - TEK4105 terminal
VT   - VT100 terminal
```

FUNCTION RESULT:

The project database project.DBF is saved in a database called project.V30. The project database is then converted to a version 4.0 database and stored in the project name project.DBF. This database is now suitable for use with all AISIM version 4.0 functions.

10.4 LUI COMMAND: MERGEIN

To move the contents of a buffer to a project database, one enters the MERGEIN command, specifying the name of the buffer and the name of the project into whose database the buffer contents are to be copied.

COMMAND SYNTAX:

MERGEIN	[PROJECT(project)]	[BUFFER(buffer)]	[TERM(terminal)]
MI	[P(project)]	[B(buffer)]	[T(terminal)]

where:

[P(project)] is a required parameter indicating the name of the project into which the entities are to be merged.

[B(buffer)] is an optional parameter indicating the name of the buffer in which the entities are stored. If omitted, the buffer is assumed to be the last buffer specified in a previous LIBRARY READY command.

[T(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

HP - HP2647A or HP2648A terminal
HP23 - HP2623A terminal
TEK - TEK4105 terminal
VT - VT100 terminal

FUNCTION RESULT:

If no entity in the buffer is the same as an entity already present in the database, the system responds:

0 CONFLICTS HAVE BEEN DETECTED IN MERGEIN INITIALIZATION

in which case the copying of the buffer contents will be completed and the user will be returned to the LIBRARY READY level. If one or more names of entities conflict with ones already in the project database, the user will be prompted with:

n CONFLICTS HAVE BEEN DETECTED IN MERGEIN INITIALIZATION

where "n" is the number of conflicts. The system then asks:

DO YOU WISH TO RESOLVE THESE CONFLICTS?

Answering "no" aborts the Mergein. If the answer is "yes", the system will then present the name of an entity which stands in conflict.

The user now has three command options to resolve the naming conflict. First, he may command that the entity in the database be deleted in favor of the one of the same name in the buffer. This is done by entering REPLACE (RP). Secondly, he may command that the entity in the buffer which aroused the naming conflict be disregarded in the transferral from the buffer to the database. This is done by issuing the command IGNORE (IG). Thirdly, one may resolve the naming conflict by giving the entity in the buffer a new name. This is done by means of the command RENAME (RN) whose one parameter is the new name the user wishes to give the entity. If the user should select as a new name one that is also being used, the system will respond with a prompt for a different name. These commands are described in detail in sections 10.4.1 through 10.4.6.

This cycle of naming conflict resolution will be repeated until all of the naming conflicts have been resolved. The system will then tell the user that MERGEIN initialization has been completed, do the MERGEIN and automatically return the user to the LIBRARY READY level.

NOTE: Resources associated with an architecture are not subject to the REPLACE command.

```
END
E

HELP

IGNORE
IG

INFO

RENAME  {name1}
RN

REPLACE
RP
```

Figure 10-4. Mergein Command Summary

10.4.1 MI COMMAND: END

The END command, issued at the MERGEIN sublevel causes the system to exit the MERGEIN sublevel and returns the user to the LIBRARY READY level.

COMMAND SYNTAX:

END

E

FUNCTION RESULT:

The system returns to the LIBRARY READY level.

10.4.2 MI COMMAND: HELP

The HELP command enables the user to obtain a menu of the commands available in the MERGEIN sublevel.

COMMAND SYNTAX:

HELP

FUNCTION RESULT:

A menu of available commands is printed on the screen.

10.4.3 MI COMMAND: IGNORE

The IGNORE command enables the user to resolve any naming conflicts encountered at the MERGEIN sublevel in favor of the entities that already exist in the target database.

COMMAND SYNTAX:

IGNORE

IG

FUNCTION RESULT:

The entity indicated by the prompt is not copied into the project database. The system then prompts the user with the next naming conflict, if any, and proceeds with MERGEIN operation.

10.4.4 MI COMMAND: INFO

The INFO command furnishes the user with information on the options available to resolve naming conflicts encountered in the MERGEIN sublevel.

COMMAND SYNTAX:

INFO

IN

FUNCTION RESULT:

The screen displays the following information:

IGNORE: THIS OPTION CAUSES THE NAMED ENTITY IN THE BUFFER TO BE EXCLUDED FROM THE MERGEIN OPERATION

RENAME: THIS OPTION CHANGES ALL OCCURANCES OF THE ENTITY NAME IN THE BUFFER TO THE NAME SPECIFIED BY THE USER

REPLACE: THIS OPTION DELETES THE NAMED ENTITY FROM THE USER DATA BASE, ALLOWING THE ENTITY IN THE BUFFER TO BE MERGED IN

END: THIS OPTION TERMINATES THE MERGEIN PRE-PROCESSING WITHOUT RESOLVING ANY MORE NAMING CONFLICTS AND RETURNS TO THE LUI READY LEVEL

10.4.5 MI COMMAND: RENAME

The RENAME command allows the user to resolve a naming conflict encountered during the MERGEIN operation by giving entities in the buffer a unique name.

COMMAND SYNTAX:

RENAME {name1}

RN

where:

{name1} is the new name the entity is to be given.

FUNCTION RESULT:

The system checks to see whether the new name given to the entity creates any naming conflicts. If it does, the system will prompt the user to that effect, and await a new name. If the new name does not create any conflicts, the entity is copied into the project database under its new name. If there are naming conflicts with further entities, the system then prompts the user for their resolution. If there are no remaining naming conflicts, the MERGEIN operation begins.

10.4.6 MI COMMAND: REPLACE

The REPLACE command enables the user to resolve a naming conflict encountered in the MERGEIN sublevel in favor of entities that exist in the buffer.

COMMAND SYNTAX:

REPLACE

RP

FUNCTION RESULT:

The entity indicated in the prompt is written into the database and the old entity of the same name is deleted. The system then proceeds to consideration of the next naming conflict if any exist. Otherwise, the MERGEIN operation begins.

10.5 LUI COMMAND: MERGEOUT

When the user wishes to place entities from a project database into a buffer, he does so via the MERGEOUT command, specifying the name of the project and the name of the buffer into which the entities are to be copied. Entities in the project are copied one at a time by name through the SELECT command. If the user needs a list of the entities of a given type, he may obtain one through the LIST command. Also available here is the HELP command which provides a menu of the other available commands. The END command will return the user to the LIBRARY READY level. These commands are described in detail in sections 10.5.1 through 10.5.4.

To obtain access to the MERGEOUT sublevel, issue the command,

MERGEOUT	[PROJECT(project)]	[BUFFER(buffer)]	[TERM(terminal)]
MO	[P(project)]	[B(buffer]	[T(terminal)]

where:

[P(project)] is a required parameter indicating the name of the project from which the entities are to be copied.

[B(buffer)] is an optional parameter indicating the name of the buffer into which the entities are to be transferred are stored. If omitted, the buffer is assumed to be the last buffer specified in a previous LIBRARY READY level command.

[T(terminal)] is an optional parameter indicating the type of terminal the user is logged on to. If omitted, the terminal type is assumed to be the last terminal type specified in a previous AISIM READY or LIBRARY READY level command. The valid terminal types are the following:

HP - HP2647A or HP2648A terminal
HP23 - HP2623 terminal
TEK - TEK4105 terminal
VT - VT100 terminal

FUNCTION RESULT:

The user is given a "*" prompt, from which he can issue one of the following commands.

1) LIST {entity-type}, to list entities in project database.

2) SELECT {entity-type},{entity-name}, to select an entity to be merged out of the project database.

3) END, which will terminate the selection of entities to be copied.

```
END
E

HELP

LIST {entity-type}
L

SELECT {entity-type},{entity-name}
S
```

Figure 10-5. Mergeout Command Summary

10.5.1 MO COMMAND: END

The END command terminates the session at the MERGEOUT sublevel and causes entities in the current project database which have been flagged by the SELECT command to be copied into the current buffer.

COMMAND SYNTAX:

END

FUNCTION RESULT:

The user will be prompted with the question:

DO YOU WANT TO LIST YOUR SELECTIONS ON THE SCREEN?

A "no" answer will cause the Mergeout procedure to take place. When all of the flagged entities have been copied into the buffer the system will return to the LIBRARY READY level.

A "yes" answer will produce a list of the entities flagged in the SELECT command. The user will then be prompted as to whether he wishes to proceed with the Mergeout operation. A "yes" answer to this second question will cause the flagged entities to be copied into the current buffer and the system will return to the LIBRARY READY level. A "no" answer will return the user immediately to the LIBRARY READY level.

10.5.2 MO COMMAND: HELP

The HELP command enables the user to obtain a menu of the other command options available in the MERGEOUT sublevel.

COMMAND SYNTAX:

HELP

FUNCTION RESULT:

A menu of available command options is printed on the screen.

10.5.3 MO COMMAND: LIST

The LIST command enables the user to obtain a list of the names of the entities of a given type that are contained in the current project.

COMMAND SYNTAX:

LIST {entity-type}

L

where:

{entity-type} is the type of entity. The valid entity types are the following:

Action	A
Constant	C
Item	I
Process	P
Queue	Q
Resource	R
Table	T
Variable	V

FUNCTION RESULT:

The screen will display a list of the names of the entities of the specified type in the current project.

10.5.4 MO COMMAND: SELECT

The SELECT command allows the user to specify which entities are to be merged out of a project database to a buffer. Scenarios and Loads cannot be selected.

COMMAND SYNTAX:

```
SELECT {entity-type},{entity-name}
S
```

where:

{entity-type} is the type of entity to be merged out. The valid entity types are the following:

Action	A
Constant	C
Item	I
Process	P
Queue	Q
Resource	R
Table	T
Variable	V

{entity-name} is the name of the entity to be merged out.

FUNCTION RESULT:

The specified entity is flagged for the Mergeout operation. The operation will take place only when the END command is issued.

SECTION 11

AISIM SIMULATION REPORTS

When a simulation is run, a number of Processes are initiated at various times throughout the simulation period. As their execution proceeds they contend for available Resources such as machines and operators. The simulation stops at the end of a predefined period and produces output statistics.

In general, any high-level performance factor measurable on a real system in terms of time, percentages, or counts of events can be measured during the model run. Experiments that are virtually impossible to run on a real system can be constructed and easily measured in the model. Specifically, measures that may be obtained are:

- Resource utilization statistics
- Total number of Processes completed
- Average elapsed time for Process completion
- System and job delays associated with actions
- Statistics on queue sizes and timing
- Variable changes during simulation
- System and job delays associated with Resources
- Execution count of Process steps

Two forms of statistical output are available to the user as a result of the simulation. Interactive output, displayed on the terminal screen, is available at any user-defined breakpoint, at the end of simulation periods, or at the end of the simulation.

The second form of output is a listing, obtained off-line, which lists the simulation measures mentioned above.

The following sections describe the simulation outputs and how to obtain them.

11.1 INTERACTIVE RESULTS AND HOW TO OBTAIN THEM

Interactive results can be viewed on the terminal while in the AUI level. A review of the AUI level shows that several commands are available for viewing data after simulation periods, after breakpoints, and after simulation termination. The DEFPLOT command is used before simulation is started to select the graphs that the user wishes to view after simulation (see the DEFPLOT command description in section 7.2 for attributes and statistics of entities that can be graphed). The LISTVAL command can be used at the points mentioned above to view simulation data concerning model entities (see the LISTVAL command description in section 7.11 for attributes and statistics of entities that can be viewed). The PLOT

command is also used at the points mentioned above to view graphically the statistics which were kept due to the DEFPLOT plot definitions. See the PLOT command definition in section 7.12 for examples of the forms and graphs that are displayed to the user as a result of this command.

11.2 REPORT RESULTS AND HOW TO OBTAIN THEM

The commands to view and print results are available at the AISIM READY level. As the simulation executes, simulation results are automatically stored in a database file named:

project.RPT

where:

project indicates that the model output report to be accessed was generated by an analyze session on the design database named PROJECT.

Two AISIM READY level commands are available to manipulate this data file. The PRINT command (see section 5.17) is used to print a listing of the simulation report at the local hardcopy facility. The EDIT command (see section 5.6) allows the user to view the project.RPT file through the use of the EDT text editor. See section 11.3 for a brief discussion of relevant EDT text editor commands. See the EDT Users Manual for additional information on the EDT text editor.

The project.RPT file contains a number of reports that describe the model that was simulated and the results of the simulation. On the following pages each of these reports is described and examples of results are given.

INITIALIZATION REPORT: This report displays the contents of the model inputs as used during this simulation. Elements of this report are:

1) Global Constant Definition
2) Table Definition
3) Global Variable Definition
4) Item Definition
5) Queue Definition
6) Resource Definition
7) Architecture Legal Path Definition
8) Action Definition
9) Process Definition
10) Load Definition
11) Scenario Definition

Figures 11-1 through 11-5 show the various parts of a typical initialization report.

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$                   S I M U L A T I O N    R E P O R T                 $
$                           AISIM VERSION 4.0                          $
$                        HUGHES AIRCRAFT COMPANY                       $
$                               02/05/85                               $
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
GLOBAL CONSTANT DEFINITION.....

CONSTANT INITIAL
MNEMONIC VALUE    COMMENT
======== ======== ====================================================
PERLNGTH 7500000

TABLE DEFINITION....


GLOBAL VARIABLE DEFINITION.....

VARIABLE INITIAL
MNEMONIC VALUE    COMMENT
======== ======== ====================================================
B-LNTH   750      750 MSG LNTH FOR HQ -> B
B-PRI    11       11 - PRIORITY OF B-ORIGIN PROCESS
BBLNTH   750      750 - LENGTH OF B NODE TO B NODE MESSAGE
BECHOPRI 11       11 - PRIORITY OF BECHO PROCESS
CHQGOVHD 28       .000280 - SEC PER WORD OF GRAPHICS OVERHEAD AT CHQ
CHQHOVHD 10       .000100 - SEC WORD PROCESSING OF HARD COPY AT CHQ
CHQLNTH  750      750*FN6*COMPRESSION = MSG LNTH HQ -> CHQ
CHQPRI   11       11 - PRIORITY OF CHQ PROCESS
GRLNTH   10000    10000 - GRAPHICS RESULT FROM CHQ TO HQ
HCLNTH   200      200 - LENGTH OF HARD COPY MESSAGE
HCPRI    11       11 - PRIORITY OF HARD COPY PROCESS
HCRLNTH  6300     6300 - LENGTH OF HARD COPY RESULT
HQGGLNTH 200      200 - LENGTH OF GRAPHICS REQUEST
HQGGPRI  11       11 - PRIORITY OF HQ HARDCOPY PROCESS
HQHGLNTH 200      200 - LENGTH OF HARDCOPY REQUEST TO CHQ
HQHGPRI  11       11 - PRIORITY OF HARDCOPY PROCESS
HQLNTH   750      750 - LENGTH OF MESSAGE SENT TO HQ NODES
HQOVHD   8        .000080 - SEC PER WORD PROCESSING AT HQ NODES
HQPRI    11       11 - PRIORITY OF HQ PROCESS
RT.OVHD  8        THIS IS A GLOBAL VALUE FOR ROUTING OVERHEAD
V.ROUTER 0        MONITOR VARIBALE TO PLOT ROUTE OVERHEAD (COMPUTED)
VD.CS    0.000001 CONTEXT SWITCHING DELTA TIME
VM.CS    0.000001 CONTEXT SWITCHING MEAN TIME
VM.ROUTE 8        0.000080 - TIME PER WORD ROUTED
VRATE    33.3     .000333 SEC TIME PER CHAR
VSPEED   0        UPDATED WITH CHANNEL SPEED FOR ALL TRANSFERS
```

Figure 11-1. Initialization Report - Constants, Tables, and Global Variables

```
ITEM DEFINITION.....

ITEM                            DESCRIPTION
======== ================== ============================================
ACK                             ACKNOWLEDGEMENT GENERATED AT COMM CENTERS BOUND FOR S
         ATTR.      INITIAL
         NAME       VALUE
         ========   ========
         HOPS       1
         LENGTH     ACKLENTH
         MESS       0
         ORIGIN     0
         RETRAN     1
         TNODE      0
         TYPE       $GOOD

ITEM                            DESCRIPTION
======== ================== ============================================
MSG04                           MESSAGES GENERATED AT S-NODES BOUND FOR COMM CENTERS
         ATTR.      INITIAL
         NAME       VALUE
         ========   ========
         ACKREC     0
         DEST1      1
         DEST2      1
         DEST3      1
         DEST4      1
         ENDTM      99999999
         ERRPROB    ERRPRB04
         HOPS       HOPS04
         LENGTH     LENGTH04
         NXTACKNM   1
         ORIGIN     S04
         RETRAN     1
         SATDELAY   DELAY04
         SNUM       S04NUM
         STARTTM    $CLOCK


QUEUE DEFINITION.....

QUEUE     MAXIMUM
MNEMONIC    SIZE    COMMENT
======== ========= ============================================================
CONTROLQ INFINITE PRIORITY ORDERED QUEUE OF RFT MESSAGES
```

Figure 11-2. Initialization Report - Items and Queues

```
RESOURCE DEFINITION.....

RESOURCE  TOTAL     INITIAL
MNEMONIC  # UNITS   # UNITS   DESCRIPTION
========  ========  ========  ========================================
B1        1         1         RESOURCE FOR NODE
          ATTR.     INITIAL
          NAME      VALUE
          ========  ========
          COST      Ø
          D.CS      VD.CS
          M.CS      VM.CS
          M.ROUTE   VM.ROUTE
          RATE      Ø

B1S1.A    1         1         RESOURCE FOR CHANNEL CONNECTOR
          ATTR.     INITIAL
          NAME      VALUE
          ========  ========
          COST      Ø
          RATE      VRATE

B1S1.B    1         1         RESOURCE FOR CHANNEL CONNECTOR
          ATTR.     INITIAL
          NAME      VALUE
          ========  ========
          COST      Ø
          RATE      VRATE

B2        1         1         RESOURCE FOR NODE
          ATTR.     INITIAL
          NAME      VALUE
          ========  ========
          COST      Ø
          D.CS      VD.CS
          M.CS      VM.CS
          M.ROUTE   VM.ROUTE
          RATE      Ø
```

```
ARCHITECTURE LEGAL PATH DEFINITION

FROM      TO        NEXT      VIA
DEVICE    DEVICE    DEVICE    LINK
========  ========  ========  ========
B1        B1        S1        B1S1.A
B1        B2        S1        B1S1.A
B1        B3        S1        B1S1.A
B1        B4        S1        B1S1.A
B1        B5        S1        B1S1.A
B1        B6        S1        B1S1.A
B1        B7        S1        B1S1.A
B1        CH        S1        B1S1.A
B1        H1        S1        B1S1.A
B1        H2        S1        B1S1.A
B1        S1        S1        B1S1.A
B1        S2        S1        B1S1.A
B1        S3        S1        B1S1.A
B1        S4        S1        B1S1.A
B1        S5        S1        B1S1.A
B1        S6        S1        B1S1.A
B1        S7        S1        B1S1.A
```

Figure 11-3. Initialization Report - Resources and Architecture Legal Path Table

```
ACTION DEFINITION.....

ACTION    ACTION
MNEMONIC  CLASS     COMMENT
========  ========  ==================================================
CHQGD.OH  MACHINE   CHQ PROCESSING OF GRAPHICS REQUEST
CHQHD.OH  MACHINE   CHQ PROCESSING OF HARD COPY REQUEST
CS.OH     CPU       PROCESSING TO PERFORM CONTEXT SWITCHING
DUMMYACT  MACHINE   ACTION TO ENABLE  CYCLIC PROGRAM CYCLES
HQ.OH     MACHINE   HQ PROCESSING OF MESSAGE
OVERHEAD  MACHINE   TIME FOR GENERAL USE
ROUTE.OH  CPU       PROCESSING DELAY TO ROUTE A MESSAGE AT NODE
XFER.OH   CHANNEL   PROCESSING DELAY TO ROUTE A MESSAGE OVER A CHANNEL

PROCESS DEFINITION.....

PROCESS
MNEMONIC            DESCRIPTION
========  ========  ==================================================
B-ORIGIN            THIS IS A B-NODE STUB PROCESS


ENTRY     OPCODE    PARM      PARM      PARM      COMMENT
========  ========  ========  ========  ========  ========================
          START     ALL       NO
          GIVEN     MSG
          RETURN    MSG
          END

LOCAL VARIABLES OF PROCESS B-ORIGIN
=====================================================================
    1 MSG       (I)
PROCESS
MNEMONIC            DESCRIPTION
========  ========  ==================================================
BECHO               THIS PROCESS ECHOES MESSAGE BACK TO ORIGINATOR


ENTRY     OPCODE    PARM      PARM      PARM      COMMENT
========  ========  ========  ========  ========  ========================
          START     ALL       NO
          GIVEN     MSG
          RETURN    MSG
          ASSIGN    MSG       FNODE               GET ORIGINATING NODE
                    TO.NODE
          CALL      MRS       WAIT      0         ROUTE RETURN MESSAGE
          GIVEN     B-ORIGIN  B-PRI     $REQNORE
                    B-LNTH    TO.NODE
          END

LOCAL VARIABLES OF PROCESS BECHO
=====================================================================
    1 MSG       (I)    2 TO.NODE          3 MRS       (P)    4 B-ORIGIN  (P)
```

Figure 11-4. Initialization Report - Actions and Processes

```
LOAD
MNEMONIC          DESCRIPTION
======== ======== ==================================================
LOADSØ7           S7 LOAD FROM BASES - B7
         LOAD     NODES
         ======== ======== ======== ========
         B7


         PROCESS           SCHEDULE
         MNEMONIC MAX #    METHOD   MEAN     DELTA    PRIORITY
         ======== ======== ======== ======== ======== ========
         DATABBØ7 125      EXPONENT 1440389           Ø
         DATABCHQ 125      EXPONENT 1440389           Ø
         DATABHQ1 125      EXPONENT 1440389           Ø
         HCOPYCHQ 125      EXPONENT 1464253           Ø

SCENARIO DEFINITION....

SCENARIO
MNEMONIC          DESCRIPTION
======== ======== ==================================================
SCENARIO          300 SECONDS PER PERIOD X 1 PERIODS = 300 SECS

PERIOD
LENGTH
========
7500000

         PERIOD   PERIOD   PERIOD   PERIOD   PERIOD   PERIOD   PERIOD
         MNEMONIC MNEMONIC MNEMONIC MNEMONIC MNEMONIC MNEMONIC MNEMONIC
======== ======== ======== ======== ======== ======== ======== ========
         1

TRIGGER  TIME  TO SCHEDULE TRIGGER  TIME  TO SCHEDULE
MNEMONIC SCHEDULE PRIORITY MNEMONIC SCHEDULE PRIORITY
======== ======== ======== ======== ======== ========
LOADSØ1  Ø        Ø        LOADSØ2  Ø        Ø
LOADSØ3  Ø        Ø        LOADSØ4  Ø        Ø
LOADSØ6  Ø        Ø        LOADSØ6  Ø        Ø
LOADSØ7  Ø        Ø        LOADHQ1  Ø        Ø
LOADHQ2  Ø        Ø

####    Ø ERRORS WERE DETECTED DURING MODEL INITIALIZATION
```

Figure 11-5. Initialization Report - Loads and Scenario

11.2.1 Constant Report

This report shows the value of the constants at simulation termination. An example of this report is shown in figure 11-6 where the labeled columns have the following significance.

CONSTANT: The name of the Constant

CURRENT VALUE: The Constant's value (in real numbers) at the end of the simulation.

```
SIMULATION TIME =        3241.00000 UNITS

CONSTANT REPORT

           CURRENT
CONSTANT   VALUE...
========   ========
BER        .00001
CCCHRATE   2.4
CHRATE01   1.2
CHRATE02   2.4
CHRATE03   2.4
CHRATE04   .6
CHRATE05   .075
CHRATE06   .075
CHRATE07   1.2
CHRATE08   .3
CHRATE09   2.4
CHRATE10   .3
CHRATE11   1.2
CHRATE12   .3
CLOCKVAL   0
DELAY01    0
DELAY02    0
DELAY03    0
DELAY04    0
DELAY05    0
DELAY06    0
```

Figure 11-6. Constant Report

11.2.2 Variable Report

Variable reports are divided into the numeric and the non-numeric variables. A sample of the report for numerical variables is shown in figure 11-7, where the columns have the following significance.

VARIABLE: The name of the Variable.

TOTAL SAMPLES: The number of times the Variable has been set to a value over the simulation period, including its initialization at the start of the simulation.

CURRENT: The value of the Variable at the end of the simulation.

MEAN: The mean of all values (including its initial value) that the Variable was set to over the simulation (i.e., the sum of the values divided by TOTAL SAMPLES).

STD DEV: The standard deviation of the values that the Variable was set to over the simulation.

MINIMUM: The minimum value that the Variable took on during the simulation.

MAXIMUM: The maximum value that the Variable took on during the simulation.

SIMULATION TIME = 7500000. UNITS

VARIABLE REPORT

NUMERIC VARIABLES...

VARIABLE	TOTAL SAMPLES	VALUE CURRENT	VALUE MEAN	VALUE STD DEV	VALUE MINIMUM	VALUE MAXIMUM
B-LNTH	1	750.	750.	0.	750.	750.
B-PRI	1	11.	11.	0.	11.	11.
BBLNTH	1	750.	750.	0.	750.	750.
BECHOPRI	1	11.	11.	0.	11.	11.
CHQGOVHD	1	28.	28.	0.	28.	28.
CHQHOVHD	1	10.	10.	0.	10.	10.
CHQLNTH	1	750.	750.	0.	750.	750.
CHQPRI	1	11.	11.	0.	11.	11.
GRLNTH	1	10000.	10000.	0.	10000.	10000.
HCLNTH	1	200.	200.	0.	200.	200.
HCPRI	1	11.	11.	0.	11.	11.
HCRLNTH	1	6300.	6300.	0.	6300.	6300.
HQGGLNTH	1	200.	200.	0.	200.	200.
HQGGPRI	1	11.	11.	0.	11.	11.
HQHGLNTH	1	200.	200.	0.	200.	200.
HQHGPRI	1	11.	11.	0.	11.	11.
HQLNTH	1	750.	750.	0.	750.	750.
HQOVHD	1	8.	8.	0.	8.	8.
HQPRI	1	11.	11.	0.	11.	11.
RT.OVHD	816	8.	8.	0.	8.	8.
V.ROUTER	1	0.	0.	0.	0.	0.

Figure 11-7. Numeric Variable Report

The report for Variables taking non-numeric values is illustrated in figure 11-8 where the labeled columns have the following significance.

VARIABLE: The name of the Variable.

CURRENT TYPE: The type of entity or construct that the Variable is set to at the end of the simulation.

CURRENT VALUE: The name of the entity or construct to which the Variable is set at the end of the simulation.

NON-NUMERIC VARIABLES...

VARIABLE	CURRENT TYPE	CURRENT VALUE
ACKST011	ALPHA	$CORRECT
STATE013	ALPHA	$CORRECT
STATE014	ALPHA	$CORRECT
STATE021	ALPHA	$CORRECT
STATE022	ALPHA	$ERROR
STATE023	ALPHA	$CORRECT
STATE024	ALPHA	$CORRECT
STATE031	ALPHA	$ERROR
STATE032	ALPHA	$CORRECT
STATE033	ALPHA	$CORRECT
STATE034	ALPHA	$CORRECT
STATE041	ALPHA	$CORRECT
STATE042	ALPHA	$CORRECT
STATE043	ALPHA	$CORRECT
STATE044	ALPHA	$CORRECT
STATE051	ALPHA	$ERROR
STATE052	ALPHA	$ERROR
STATE053	ALPHA	$ERROR
STATE054	ALPHA	$ERROR
STATE061	ALPHA	$ERROR
STATE062	ALPHA	$ERROR
STATE063	ALPHA	$ERROR
STATE064	ALPHA	$ERROR
STATE071	ALPHA	$CORRECT
STATE072	ALPHA	$CORRECT
STATE073	ALPHA	$CORRECT
STATE074	ALPHA	$CORRECT
VAR	RESOURCE	CPU

Figure 11-8. Non-numeric Variable Report

11.2.3 Item Report

Figure 11-9 illustrates the Item Report, where the labeled columns have the following significance.

ITEM NAME: The name of the Item.

NUMBER CREATED: The number of instances of this Item that have been created with the CREATE or SEND Primitives over the simulation.

NUMBER DESTR'D: The number of instances of this Item that have been destroyed with the DESTROY Primitive over the simulation.

TIME IN SYSTEM - MINIMUM: The minimum time any instance of the Item was in the system.

TIME IN SYSTEM - MAXIMUM: The maximum time any instance of the Item was in the system.

TIME IN SYSTEM - AVERAGE: The average time any instance of the Item was in the system.

TIME IN SYSTEM - STD DEV: The standard deviation in the times the Item spent in the system.

MINIMUM, MAXIMUM, AVERAGE, STD DEV are based on the individual Item instances' time in the system. This statistic is calculated whenever an Item instance is destroyed (with the DESTROY Primitive) and is equal to the time of destruction minus the time of creation (with the CREATE or SEND Primitive). Therefore, Items in the system that have not been destroyed at simulation end will not be reflected in these statistics.

SIMULATION TIME = 3241.0000 UNITS

ITEM REPORT

ITEM NAME	NUMBER CREATED	NUMBER DESTR'D	TIME IN SYSTEM MINIMUM...	MAXIMUM...	AVERAGE...	STD DEV...
RFT1MSG	67	45	89.09	1015.92	557.62	232.58
RFT2MSG	54	37	52.47	1030.96	654.01	253.07
RFT3MSG	56	41	106.29	1024.44	644.62	237.49
RFT4MSG	51	33	180.87	1055.10	658.14	238.19
RFT5MSG	52	39	180.34	964.16	612.76	231.33

Figure 11-9. Item Report

11.2.4 Resource Report

This report gives statistics on each Resource's presence in the idle state, busy queue, and inactive state as well as the number of Processes put into a wait queue for the Resource. These queues are discussed in detail in the section on system defined queues (see section 3.5). Four kinds of statistics are kept on the busy and wait queues: (1) entities put into the queue (INTO), (2) entities taken out of a queue (OUT OF), (3) the number in the queue (#), and (4) the time entities spent in the queue (TIME). Statistics on the number in the state are kept for the idle and inactive states.

An example of the Resource Report on these states and queues is shown in figure 11-10. For each row of each queue or state the numbers have the following significance.

The TOTAL NUMBER of the INTO and OUT OF rows indicate the number of entities that were, respectively, placed in or taken out of the queue.

The CURRENT # is the number of entities in the queue or state at the time the simulation run was completed.

The MEAN # is the time weighted average of the number of entities in the queue or state over the simulation.

The STD DEV # is the standard deviation in the number of entities in the queue or state over the simulation.

The MINIMUM # is the minimum number of entities in the queue or state at one time over the simulation.

The MAXIMUM # is the maximum number of entities in the queue or state at one time over the simulation.

The MEAN TIME is the average time entities spent on the queue.

The STD DEV TIME is the standard deviation in the time that the entities spent on queue.

The MINIMUM TIME is the minimum time any entity was in the queue.

The MAXIMUM TIME is the maximum time any entity was in the queue.

The REQUEST TIME statistics provide the mean, standard deviation, minimum and maximum of the time it took for the request for each unit of the Resource to be satisfied. I.e., the request time is the difference between the time an allocate request is made and the time the Resource unit is placed in the busy queue.

The field labeled "CURRENTLY ALLOCATED TO PROCESSES:" provides a list of the Processes whose task instances had allocated the Resource at simulation end.

The field labeled "PROCESSES CURRENTLY WAITING:" provides a list of the Process task instances which were suspended while waiting for the Resource at the end of the simulation.

```
SIMULATION TIME =     7500000.       UNITS

RESOURCE REPORT

                    TOTAL
RESOURCE            NUMBER    CURRENT...  MEAN.....  STD DEV...  MINIMUM...  MAXIMUM...
================    ========  ==========  ==========  ==========  ==========  ==========
B1
          # IDLE                      1.        .941        .236          0.          1.
    REQUEST TIME                             577.941    2431.301          0.      14013.

       INTO BUSY          34
     OUT OF BUSY          34
          # BUSY                      0.        .059        .236          0.          1.
       BUSY TIME                           13058.826   17363.084     1600.      50400.063

      # INACTIVE                      0.          0.          0.          0.          0.

       INTO WAIT          34
     OUT OF WAIT          34
       # WAITING                      0.        .003        .051          0.          1.
       WAIT TIME                            577.936    2431.295          0.      14013.

     CURRENTLY ALLOCATED
           TO PROCESSES: NONE

     PROCESSES CURRENTLY
                WAITING: NONE
```

Figure 11-10. Resource Report

11.2.5 Action Report

The Action Report provides the user with statistics on the time consumed by each Action. Statistics are gathered on two aspects of such time consumption, called "useful time" and "delay time".

"Useful time" is equal to the amount of time the Action was being executed, whereas "delay time" is the time between the initiation and completion of an Action during which the execution of the Action (i.e., the Process in which it appears) is suspended. Both useful time and delay time are calculated only upon the completion of the Action. Therefore, Actions which are active at the end of the simulation are not included in these statistics.

A sample Action Report is shown in figure 11-11. The name immediately below the ACTION heading is the user-defined name of the Action. For the row labeled USEFUL TIME the statistics have the following significance:

TOTAL SAMPLES: the number of times the useful time was calculated (i.e., the number of times the Action was completed).

MEAN: The average useful time of this Action over the simulation (i.e., the total time taken by the Action divided by TOTAL SAMPLES).

STD DEV: The standard deviation in the useful times.

MINIMUM: The minimum time taken in the execution of the Action over the simulation.

MAXIMUM: The maximum time taken in the execution of the Action over the simulation.

% TIME OF TOTAL: The percent of the total simulation time for which this Action was executing. Since AISIM allows for the parallel execution of the same Action, this figure can be greater than 100.

The figures in the row labeled DELAY TIME have the following significance.

TOTAL SAMPLES: The number of times the delay time was calculated (i.e., the number of times the Action was completed). This will always be equal to the TOTAL SAMPLES of USEFUL TIME.

MEAN: The average time the Action was delayed during execution over the simulation (i.e., the total time taken up in delay divided by TOTAL SAMPLES).

STD DEV: The standard deviation in the delay times over the simulation.

MINIMUM: The minimum delay time of an Action over the simulation.

MAXIMUM: The maximum delay time of an Action over the simulation.

Note that % OF TOTAL is not calculated for the delay time.

SIMULATION TIME = 7500000. UNITS

ACTION REPORT

ACTION	TOTAL SAMPLES	MEAN.....	STD DEV...	MINIMUM..	MAXIMUM...	% TIME OF TOTAL.
CHQGD.OH						
USEFUL TIME	51	109803.923	89251.581	62999.969	280000.063	74.667
DELAY TIME	51	0.	0.	0.	0.	

ACTION	TOTAL SAMPLES	MEAN.....	STD DEV...	MINIMUM..	MAXIMUM...	% TIME OF TOTAL.
HQ.OH						
USEFUL TIME	78	6000.	0.	6000.	6000.	6.240
DELAY TIME	78	153.846	1349.995	0.	12000.	

ACTION	TOTAL SAMPLES	MEAN.....	STD DEV...	MINIMUM..	MAXIMUM...	% TIME OF TOTAL.
ROUTE.OH						
USEFUL TIME	811	14601.233	21302.749	1599.969	80000.	157.888
DELAY TIME	811	0.	0.	0.	0.	

ACTION	TOTAL SAMPLES	MEAN.....	STD DEV...	MINIMUM..	MAXIMUM...	% TIME OF TOTAL.
XFER.OH						
USEFUL TIME	639	50953.905	82729.325	700.	417000.	434.127
DELAY TIME	639	0.	0.	0.	0.	

Figure 11-11. Action Report

11.2.6 Queue Report

The Queue Report provides statistics on the utilization of user defined Queues. The report contains information both on the number of entities stored on the Queue as well as information on the impact the utilization of the Queue had on Process execution and suspension. A sample Queue Report is shown in figure 11-12. The rows labeled FILED ON, REMOVED FROM, # IN QUEUE and TIME IN QUEUE key statistics on the manipulation of the Queue itself. The rows labeled TASKS BLOCKED, TASKS RESUMED, # BEING BLOCKED, TIME BLOCKED refer to statistics on Process tasks that have been suspended because they attempted to file an entity on a Queue that was full (i.e., whose maximum number had been exceeded.)

The statistics in each category have the following significance.

The TOTAL NUMBER/FILED ON is the number of entities that have been filed on the Queue over the whole simulation.

The TOTAL NUMBER/REMOVED FROM is the number of entities that have been removed from the Queue over the simulation.

The CURRENT/# IN QUEUE is the number of entities on the Queue at the time of simulation end.

The MEAN/# IN QUEUE is the time weighted average of the number of entities on the Queue over the simulation.

The STD DEV/# IN QUEUE is the standard deviation in the number of entities on the Queue over the simulation.

The MINIMUM/# IN QUEUE is the minimum number of entities on the Queue at any time during the simulation (this statistic is always zero since the Queue will be empty at the start of the simulation).

The MAXIMUM/# IN QUEUE is the maximum number of entities residing on the Queue at any time during the simulation.

The MEAN/TIME IN QUEUE is the average time entities spent on the Queue.

The STD DEV/TIME IN QUEUE is the standard deviation of the in times entities spent on the Queue.

The MINIMUM/TIME IN QUEUE is the least amount of time any entity spent on the Queue.

The MAXIMUM/TIME IN QUEUE is the greatest amount of time any entity spent on the Queue.

The statistics on the blocking of tasks due to the filling of Queues have the following significance.

The TOTAL NUMBER/TASKS BLOCK is the number of Process tasks that were suspended over the simulation due to Queue blocking.

The TOTAL NUMBER/TASKS RESUMED is the number of Process tasks resumed after having been blocked due to the filling of a Queue.

The CURRENT/# BEING BLOCKED is the number of Process tasks blocked at the time of simulation end.

The MEAN/# BEING BLOCKED is the average of the number of Process tasks being blocked over the simulation.

The STD DEV/# BEING BLOCKED is the standard deviation in the number of tasks being blocked over the simulation.

The MINIMUM/# BEING BLOCKED is the fewest number of Process tasks blocked at any time during the simulation.

The MAXIMUM/# BEING BLOCKED is the greatest number of Process tasks blocked at any time during the simulation.

The MEAN/TIME BLOCKED is the average of the times Process tasks were blocked during the simulation.

The STD DEV/TIME BLOCKED is the standard deviation in the times Process tasks were blocked during the simulation.

The MINIMUM/TIME BLOCKED is the least amount of time a Process task was blocked during the simulation.

The MAXIMUM/TIME BLOCKED is the greatest amount of time a Process task was blocked during the simulation.

QUEUE REPORT

QUEUE	TOTAL NUMBER	CURRENT...	MEAN.....	STD DEV...	MINIMUM...	MAXIMUM...
CONTROLQ						
FILED ON	273					
REMOVED FROM	273					
# IN QUEUE		0.	4.569	3.477	0.	13.000
TIME IN QUEUE			54.237	30.619	8.099	102.600
TASKS BLOCKED	0					
TASKS RESUMED	0					
# BEING BLOCKED		0.	0.	0.	0.	0.
TIME BLOCKED			0.	0.	0.	0.

Figure 11-12. Queue Report

11.2.7 Process Report

This report gives information on all aspects of Process executions. As mentioned before, Processes contend for Resources and many times must wait for another Process to complete before the current Process completes. Times spent in these states as well as other important data are recorded automatically for the user.

The Process Report provides the following statistics:

1) TOTAL SAMPLES - the number of times the Process was initiated, the total (overall Process instances) number of times the Process waited for another Process to complete and for required Resources to become available.

2) The sum total of time spent in all executions of this Process, sum total of waits on Processes and also Resources.

3) The mean time required for execution of the Process, for waiting on Processes, for waiting on Resources.

4) The standard deviation of time the Process required for execution, for waiting on Processes, for waiting on Resources.

5) The minimum time required for Process execution, minimum time spent waiting for other Processes, minimum time spent waiting for Resources.

6) The maximum time required for Process execution, maximum time spent waiting for other Processes, maximum time spent waiting for Resources.

7) Total number of times this Process was scheduled to execute.

8) The number of times this Process was scheduled to execute by a Load or Scenario.

9) The number of times this Process was scheduled to execute due to a call from another Process.

10) The total number of times this Process completed execution.

11) The total number of times this Process did not complete execution.

12) Total number of times the execution of this Process was suspended during execution.

13) Names of Items used in this Process.

14) Number of each Item created by this Process.

15) Number of each Item passed to this Process via the SEND Primitive.

16) Number of each Item passed out of this Process via the SEND Primitive.

17) Number of each Item destroyed by this Process.

18) Total number of each Item used in this Process.

19) Mean time each Item was held by this Process.

20) Minimum time an Item was held by this Process.

21) Maximum time an Item was held by this Process.

22) Standard deviation of time an Item was held by this Process.

23) Verbal description of the Process.

24) How many times each Primitive in the Process was executed.

25) Any entry Primitives and their names.

26) Names of other Primitives in this Process.

27) Any parameters or Items associated with each Primitive in the Process.

28) Any comment associated with each Primitive in the Process.

An example of a Process Report is shown in figure 11-13.

```
                    TOTAL
PROCESS             SAMPLES  SUM . . .  MEAN . . .  STD DEV. . . MINIMUM . . MAXIMUM
TRANSMIT
          TOTAL         194  121608 022    626.845     216.331     178 875     968 625
   PROCESS WAIT           0         0          0.          0.          0.           0
  RESOURCE WAIT           0         0          0.          0.          0.           0

        TOTAL #  # AUTO   # CALL   # OF      # NOT     # TIMES
        SCHEDULE SCHEDULE SCHEDULE COMPLETE  COMPLETE  SUSPEND
            279        0      279      272         7         0

         ITEM      CREATED  RECEIVED SENT     DESTR'D
         RFT1MSG         0         0        0      51
         RFT2MSG         0         0        0      61
         RFT3MSG         0         0        0      47
         RFT4MSG         0         0        0      54
         RFT5MSG         0         0        0      60

                    PROCESS  HOLDING  TIME
         ITEM       # SMPLS  MEAN. . .  MINIMUM. .  MAXIMUM.  STD DEV
         RFT1MSG         51       6  2         0.      33 75       8.18
         RFT2MSG         61      17 37         0       40 50      11 23
         RFT3MSG         47      26 57         0       47 25      10 82
         RFT4MSG         54      41 62      13 50       67 50      13 48
         RFT5MSG         60      54 11      27 00       81 00      14 08

PROCESS             DESCRIPTION
TRANSMIT            TRANSMIT THE MESSAGES

COUNT ENTRY    OPCODE    PARM      PARM      PARM     COMMENT
  279          START               NO
  279          GIVEN     LMSG      MSGLGTH   TIME
  279          MTTIME    CONSTANT  TIME               WAIT FOR MT SLOTS TO ARIVE
  273          DESTROY   LMSG                         STOP TIME IN SYSTEM CLOCK
  273          MTUSE     CONSTANT  3 375              WAIT TILL MIDDLE OF MTSLOT
  272          EVAL      TSLOTUSD  ADD                CREDIT USE OF FIRST MTSLOT
  272                    TSLOTUSD  1
  272 NEWSLOT  ENTRY
  272          EVAL      MSGLGTH   SUBTRACT           CREDIT TRANS OF AN MTSLOT
  272                    MSGLGTH   1
  272          COMPARE   MSGLGTH             EQ       TRANSMISSION FINISHED?
  272                    0                   END
    0          MTUSE     CONSTANT  6 75               WAIT TILL MIDDLE OF MTSLOT
    0          EVAL      TSLOTUSD  ADD
    0                    TSLOTUSD  1
    0          BRANCH    NEWSLOT   100
  272 END      ENTRY
  272          END
```

Figure 11-13. Process Report

11.3 COMMANDS RELEVANT TO VIEWING OUTPUT REPORTS

To view output reports of simulation runs of a model from the AISIM READY level, one uses the EDIT command.

Since the output report is too long to fit on a terminal screen, to view it all, one must use some text editing commands. Below is a brief review of the commands that are most useful for this purpose. (This discussion refers to the VAX/VMS EDT text editor).

Note: In the following commands "." represents the current line in the file.

11.3.1 TOP, BOTTOM

To orient the screen to either the top or bottom of the report one should enter one of these two commands.

```
TYPE   BEGIN
TYPE   END
```

11.3.2 UP, DOWN

To move the report either up or down on the screen n lines issue the command,

```
TYPE .-n
```

or

```
TYPE .+n
```

and the line n lines up or down from the current one will be printed.

11.3.3 FIND

To find a certain sequence of characters, sequence, enter the characters between delimiting single quotes.

```
TYPE 'SEQUENCE'
```

and the screen will print the nearest line down in the text containing the characters sequence.

11.3.4 LIST

To print n consecutive lines down from the one to which one is currently oriented, issues the command,

```
TYPE .:.+n
```

and the next n lines will be displayed on the screen.

APPENDIX A

OPERATIONAL PROCEDURES AND IMPORTANT INFORMATION

A.1 IMPORTANCE OF DATABASE BACKUP AND ALLOCATION

Processes and the other model entities are stored on disk as they are input to AISIM. Changes and additions made to this information are reflected in the current version of the database on disk. It is possible for this database to be damaged if the computer system fails or if the input session is abnormally terminated while a change or addition is being made so that it is unusable. In addition, errors made in inputting may make the stored information nonsensical if they are severe enough. For these reasons, the BACKUP command is provided.

It is wise to periodically create a backup copy of the database with the AISIM READY level command "BACKUP". Should a database be damaged, it may be recreated from the last BACKUP copy by using the "RESTORE" command.

A.2 ABNORMAL TERMINATION OF A DUI OR AUI SESSION

To terminate a DUI or AUI session normally the user must enter the command END. If the user becomes entwined in a situation which disallows normal system operation, the following procedures should be followed:

It should be noted that while in a DUI session, only the data entered prior to the last SAVE command will remain intact after this procedure is executed. If the system appears to malfunction, caution should be used in issuing a SAVE command. If the database is the source of the malfunction and a SAVE command is issued, the user might destroy the entire database. It is better to lose one session's data (by not saving) than to destroy an entire database.

> If the user is on an HP terminal, strike the TERMINAL RESET key until the message "TERMINAL READY" appears in the upper left hand corner of the screen; two strikes in a one-second period are required.
>
> Then on any terminal, type the cntl (control) key and the C key simultaneously.

If no response to these procedures is seen, the user should disconnect the modem, and try to log in and reinitiate AISIM.

If the system responds by displaying "$" the user should reinvoke AISIM.

A.3 AISIM PLOTS

The following section is intended to describe in detail how the simulation plot results produced by the AISIM Analysis function are generated. This discussion addresses the implementation of the plot function in AISIM with respect to the physical characteristics of the terminal display and the driving software. For a user of AISIM, it is generally not necessary to be aware of implementation specific details. This section has been included because the plot output from AISIM simulation runs is the most visible form of output produced. This data may appear to contradict other results produced by the AISIM Analyze function (output listing statistics). This explanation is intended to describe how this function works so that the AISIM user can explain apparent anomalies.

AISIM produces plotted data for many statistics. The plots represent "instantaneous" output from the simulation because in all cases, a defined statistic is plotted against time (the y-axis is the statistic value, the x-axis is the simulation clock). Time is normally considered to be continuous; therefore, it is "reasonable" to assume that AISIM plots are continuous. In reality, this is not the case. AISIM plots are produced by sampling statistics at discrete intervals during the simulation. Each sample defines a point on the plot. A couple of relationships need to be known to understand how this sampling technique produces plots.

The first relationship a user must be aware of is the resolution of the display screen. The terminal graphics terminals have a raster scan display. A raster is the smallest addressable unit which can be illuminated on the screen. Within the AISIM plot axis there are a fixed number of rasters along the x-axis (700 for the HP terminals, 500 for TEK4105, and 1024 for VT100). What this implies is that up to a fixed number of points can be plotted along the x-axis without exceeding the hardware limitations of the display. When an AISIM user specified a plot be displayed which has more than a fixed number of points, the AISIM software reduces the data sent to the terminal so that it can be displayed. This data reduction has the effect of "ignoring" some points. When points are ignored, the obvious result is that the plots lose accuracy. This can account for discrepancies between the plotted data and the simulation summary results, specifically with respect to the minimum and maximum statistics. The simulation report may indicate that a Resource queue had a maximum length of 100 when a plot of the current number in wait for a Resource over time indicates only a maximum value of 80.

Another problem which can occur with respect to plotting is that the plot sampling can miss activity occurring in the simulation because the sample interval is too long. The following default relationship is embedded in the AISIM software. One hundred data points are sampled for each period in the Scenario definition of a simulation run.

What this implies is that if a Scenario is defined to have only one period, only one hundred plot samples will be collected. The sample interval is calculated as the period length/100.0. Suppose the period length is defined to be 3000 units (where units are seconds, this is 1 hour). Plot samples are collected every 36 units (or 36 seconds). If

activity occurs in the model over time intervals less that 36 units, this data will not be captured for plotting. This could occur if a user wanted to see a plot of disk utilization of a computer system over a one-hour time frame. Since disk operations occur in seconds or less, a plot of the current number busy of the Resource disk would miss most of the data points if samples were taken every 36 seconds.

It is possible to adjust the plot sampling interval in the Scenario definition. The number of samples collected for each plot is computed as the number of periods in the Scenario multiplied by 100 points.

To reiterate, AISIM plots produce graphs of statistics collected during a simulation run, and display the results over time. The data for these plots is collected by sampling discrete intervals. It is not generated by state changes detected by the simulator. Therefore, the "instantaneous" plots of "CURRENT" data over time can disagree with accumulated statistics in the simulation listing.

A.4 PRODUCING HARDCOPIES OF THE TERMINAL DISPLAY

In addition to producing hardcopies of the Process flowcharts, the HP2631G Graphics Printer, the TEK4695 copier, or the HP2623 internal printer can be used to produce hardcopies of the architecture, plots, or Process diagrams.

The user is warned especially against copying forms on the TEK4105 terminal since this action will empty the ink wells on the TEK4695 copier. The interfaces on a TEK4105 terminal define the screen to be a dark blue color, so attempts to copy the forms screen will cause a page full of blue ink.

To produce hardcopies of the terminal display of an HP2647A terminal, the following must be in effect:

1) An HP2631G Graphics Printer must be connected to the HP2647A Graphics Terminal with the HP-IB communications bus.

2) The HP-IB bus address of the printer must be set to one.

3) The printer must be set to ON LINE mode.

To transfer the display information to the printer, the user first presses the <COMMAND> key. This places the terminal in "command mode".

To transfer text (e.g., Plot titles, LISTVAL responses), the user then presses the following keys in succession: <F1> <F1> <F3> <F3> <F3> <F7> <1> <RETURN>.

To transfer graphics (e.g., Architecture displays, plots), the user presses the following keys in succession: <F1> <F1> <F3> <F3> <F4> <F7> <1> <RETURN>.

NOTE: Any text preceeding the cursor position will not be transferred. Thus, the user should be sure the cursor is placed in the proper position before placing the terminal in "command mode".

When the transfer process is complete, the user exits the "command mode" by once again pressing the <COMMAND> key.

If the user is on a TEK4105 terminal equipped with a TEK4695 printer, the SCOPY button will copy any data on the screen from the terminal to the printer.

The user can print the smaller size copies by using the following procedure before the copy is made:

1. Press the SETUP key (an asterisk should appear).

2. Type HCSIZE 1

3. Press the SETUP key again

4. Perform the copy

The terminal can be reset for normal copy size by following the above procedure and typing a zero instead of a one in line 2.

If the user is on a HP2623 terminal, the following keys will cause any data on the screen to be copied to the internal printer:

<modes> - display terminal modes
<remote> - set terminal off line
<enter key> - perform copy
<remote> - set terminal back on line

A.5 EXECUTING SIMULATION RUNS AS BATCH JOBS

Once a developed model has been translated, it is not necessary to execute simulation runs interactively. They may be executed as batch jobs. The advantages of the batch method are:

1) The user does not have to remain at the terminal through out the AISIM session. All necessary job information is specified up front and the system takes charge.

2) It is not necessary to use a graphics terminal. Any terminal connected to the VAX will suffice.

3) Multiple simulation runs can execute concurrently.

4) Simulation runs can be deferred to execute during off-peak hours.

To set up a batch execution, the user types "BATCH" at the AISIM READY level. The system then prompts the user for the following information.

ENTER NAME OF PROJECT (1-8 char): the name of the project to be used.

DO YOU WISH TO TRANSLATE THE MODEL? yes or no based on the user's choice.

ENTER COMMANDS FOR AISIM RUN (<CR> TO END)
>

Enter commands for AISIM run. Allowable AUI commands are CANBREAK, DELETE, EDIT, END, GET DEF, GO, INFRES and SAVE.

Commands are typed one per line, in the order they are to be acted upon. Commands must be typed in the correct format. The GO and END commands are mandatory. All other commands are optional.

After the above processing is completed, a file called SUBBATCH.COM will have been created. This file can then be submitted to an appropriate batch queue with any other information such as at what time the job should run (see VAX SUBMIT command for available parameters). If no extra information is necessary, the following command will submit the AISIM job to the default batch queue to be run immediately:

```
SUBMIT SUBBATCH.COM
```

Figures A-1 and A-2 show sample batch run setups.

```
AISIM READY
>BATCH
ENTER NAME OF PROJECT (1-9 CHAR): project
DO YOU WISH TO TRANSLATE THE MODEL? yes
ENTER COMMANDS FOR AISIM RUN (<CR> TO END)
>e c,msgarate,1
>e c,errorate,2
>go
>end
>
SUBBATCH.COM CREATED
AISIM READY
>submit subbatch.com
  Job 303 entered on queue SYS$BATCH
AISIM READY
```

Figure A-1. Sample Batch Job Submission

```
AISIM READY
>batch
ENTER NAME OF PROJECT (1-8 CHAR): project
DO YOU WISH TO TRANSLATE THE MODEL? yes
ENTER COMMANDS FOR AISIM RUN (<CR> TO END)
>e c,msgarate,1
>e c,errorate,2
>get def,plotdef
>go
>save plot,plots,plots created in simulation run
>end
>no
>
SUBBATCH.COM CREATED
AISIM READY
>submit subbatch.com
  Job 304 entered on queue SYS$BATCH
AISIM READY
```

Figure A-2. Sample Batch Job Submission with Plots

A.6 RANDOMNESS IN RESULTS

There are ten random number streams available for use by the functions producing the random results associated with Loads, probabilistic branching (with the PROB Primitive), and Action durations.

For the Load entity, the random number stream is used by the probability functions that determine the time between Process triggerings. For the PROB Primitive, the random number stream is used in evaluating whether or not execution should branch to the given point. For the ACTION Primitive, the random number stream is used by the probability functions that determine the duration of an Action.

The user may select the random number stream used by each of these three functions using the EDIT command (see section 7.4) in the AUI. The default values are one, two, three, for Loads, PROB Primitives, and ACTION Primitives, respectively. The current stream assignments can be displayed with the LISTVAL command (see section 7.11) in the AUI.

When simulating a system, the user needs to have a sufficient number of observations to analyze in order to draw valid conclusions. It is sometimes desirable to execute additional simulation runs with the same conditions to obtain additional observations. To do this, the random number streams should be changed for each additional run. Otherwise, the results will not change.

APPENDIX B

AISIM ERRORS

If there are errors detected during the initialization, an error message will be written below the invalid entry. Following is a list of the initialization error messages and their causes.

ERROR - VALUE MUST BE NUMERIC

A non-numeric value was found as the value of a Constant. The defined value of a Constant must be numeric.

ERROR - TABLE ENTRIES MUST BE NUMERIC

A non-numeric value was found as an entry in a D or C type Table. All D or C type Table entries must be numeric.

ERROR - ALPHA TABLE X ENTRY IS ILLEGAL TYPE

In an alpha Table, an x entry was a Keyword or other invalid entry. The only valid entries are references to Actions, Items, Processes, Queues, Resources, or Tables.

ERROR - ALPHA TABLE Y ENTRY IS ILLEGAL TYPE

In an alpha Table, a y entry was a Keyword or other invalid entry. The only valid entries are references to Actions, Items, Processes, Queues, Resources, or Tables.

ERROR - VARIABLE INITIALIZED TO ILLEGAL TYPE

A Keyword or other illegal type was found as the value of a variable. Variables must be initialized to Actions, Processes, Queues, Resources, Tables, Alpha Literals, or numerics.

ERROR - ATTRIBUTE DEFINED MORE THAN ONCE

An Item, Process, or Resource attribute was defined more than once. The duplicate attribute definition should be removed.

ERROR - ******** NOT DEFINED AS A GLOBAL CONSTANT

A non-numeric value in the size field of a QUEUE was not defined as a global Constant. A non-numeric value for the size must either be the word "INFINITE" or be a previously defined global Constant.

A non-numeric value in the total or initial units field of a Resource was not defined as a global Constant. The total and initial units of a Resource must each be either a numeric value or be a previously defined global Constant.

In the definition of a Scenario, a non-numeric value in the schedule field was not defined as a global Constant. The schedule must be a numeric value or a defined Constant.

In the definition of a Scenario, a non-numeric value in the priority field was not defined as a global Constant. The priority must be a numeric value or a defined Constant.

ERROR - INITIAL # OF RESOURCE UNITS IS GREATER THAN TOTAL # OF UNITS

In a Resource definition, the initial number of units defined was greater than the total number of units of that Resource which were to be made available.

ERROR - FROM NODE IS NOT DEFINED AS A RESOURCE

In an entry in the Legal Path Table, the node specified in the FROM NODE column was not the name of a defined Resource.

ERROR - TO NODE IS NOT DEFINED AS A RESOURCE

In an entry in the Legal Path Table, the node specified in the TO NODE column was not the name of a defined Resource.

ERROR - NEXT NODE IS NOT DEFINED AS A RESOURCE

In an entry in the Legal Path Table, the node specified in the NEXT NODE column was not the name of a defined Resource.

ERROR - LINK IS NOT DEFINED AS A RESOURCE

In an entry in the Legal Path Table, the link specified in the VIA LINK column was not the name of a defined Resource.

ERROR - LABEL MUST START IN COLUMN 1 OR OPCODE MUST START IN COLUMN 10

In a Process definition, a value was encountered which did not start in column 1 or in column 10. If the value is a label, it must start in column 1, or if it is an opcode, it must start in column 10.

ERROR - OPCODE MUST START IN COLUMN 10

In a process definition, a non-label value was encountered which did not start in column 10. All opcodes must start in column 10.

ERROR - ******** NODE NAME IS NOT RECOGNIZED AS A RESOURCE

An invalid value was encountered in the node field of a Process definition. This field must be blank, contain the word "ALL", or contain a value which resolves to the name of a defined Resource.

ERROR - ******** NAME IN GIVENS LIST IS IN ERROR IN THIS CONTEXT

GLOBAL NAMES, NUMBERS AND CLOCK CANNOT BE GIVEN

The value of a given parameter for the START figure of a Process was either a numeric value or the CLOCK. Numeric values and the CLOCK cannot be used as given parameters in a Process.

ERROR - ******** ITEM IN RECEIVES LIST IS IN ERROR

This is a general message indicating an error in a START figure of type "ITEM" of a Process. This message is generally followed by one of the two following messages which more specifically describe the error.

ERROR - ITEM APPEARS TWICE IN RECEIVES LIST

In the definition of a START Primitive of type "Item," an Item was listed more than once. An Item should only occur once in the receives list of the START Primitive.

ERROR - REFERENCE IN RECEIVES LIST IS NOT DEFINED AS AN ITEM

In the definition of a START Primitive of type "ITEM," a value which was listed in the receives list was not defined as an Item. A Process with an ITEM START can only receive Items.

ERROR - ******** NUMERIC REFERENCE IN CALL PROCESS FIELD

In the definition of a CALL Primitive in a Process, the process name field contained a numeric value or a keyword. This field must contain the name of a defined Process to be initiated.

ERROR - RETURN PARAMETERS NOT ALLOWED FOR CALL NOWAIT OR BLOCK

In the definition of a CALL Primitive in a Process, return parameters were defined, but the CALL option was defined as NOWAIT or BLOCK. Only Processes called with a WAIT option can return parameters.

ERROR - ******** NUMERIC OR GLOBAL MAY NOT BE USED AS RETURN

In the definition of a CALL Primitive in a Process, a numeric value, keyword, or the CLOCK was defined as a return parameter. Numeric values, keywords and the CLOCK cannot be used as return parameters.

ERROR - BRANCH CONTINUATION DOES NOT FOLLOW A BRANCH STATEMENT

In the definition of a BRANCH Primitive of a Process, the label to branch to was not given. A branch Primitive must include a label to branch to.

ERROR - KEYWORD CANNOT BE USED IN PROB

In the definition of a probabilistic BRANCH Primitive of a Process, CLOCK or a keyword was used as the probability of BRANCH. These cannot be used as the BRANCH probability. Valid values for the BRANCH probability are numeric values and local and global Variables and Constants.

ERROR - ******** CHECK REFERENCE MUST BE RESOURCE OR QUEUE

In the definition of a TEST Primitive in a Process, the value to be tested was defined as a numeric, a global Variable, or a global Constant. The value to be tested must be a reference to either a Resource or Queue.

ERROR - ******** NUMERIC REFERENCE INVALID IN RESOURCE FIELD

In the definition of a RESET Primitive in a Process, the value to be reset was a reference to a numeric value. The value to be reset must be a reference to a defined Resource whose allocation is to be changed.

In the definition of an ALLOC Primitive in a process, the value in the name field was a reference to a numeric value. The value in the name field must be the name of a reference to a defined Resource which is to be allocated.

In the definition of a DEALLOC Primitive in a Process, the value in the name field was a reference to a numeric value. The value in the name field must be the name of a reference to a defined Resource which is to be deallocated.

ERROR - ******** REFERENCE INVALID IN ALLOCATION TYPE FIELD

In the definition of an ALLOC Primitive in a Process, the value in the allocation type field was invalid. The valid entries are "PARTIAL" and "ALL".

ERROR - BRANCH LABEL ******** NOT DEFINED IN OFD

In the definition of a Process, a BRANCH Primitive referenced a label for which there was no corresponding ENTRY label defined. An ENTRY Primitive must be used to define the label to be BRANCHed to.

ERROR - LOOP LABEL ******** NOT DEFINED IN PROCESS

In the definition of a Process, a LOOP Primitive referenced a label for which there was no corresponding ENTRY label defined. An ENTRY Primitive must be used to define the label to be branched to.

ERROR - CHECK LABEL ******** NOT DEFINED IN PROCESS

In the definition of a Process, a TEST Primitive referenced a label for which there was no corresponding ENTRY label defined. An ENTRY Primitive must be used to define the label to be branched to.

ERROR - COMPARE LABEL ******** NOT DEFINED IN PROCESS

In the definition of a Process, a COMPARE Primitive referenced a label for which there was no corresponding ENTRY label defined. An ENTRY Primitive must be used to define the label to be branched to.

ERROR - ******** ALREADY DEFINED AS AN ENTRY NAME IN THIS PROCESS

In a Process definition, an ENTRY Primitive was defined twice with the same label. A label can occur only once in a Process.

ERROR - '********' KEYWORD CANNOT BE ASSIGNED NEW VALUE

In the definition of an ASSIGN Primitive in a Process, an attempt was made to assign a new value to a Keyword other than $CNODE. Only the $CNODE keyword can be assigned a new value.

ERROR - NUMERIC QUANTITY CANNOT BE ASSIGNED A VALUE

In an ASSIGN Primitive of a Process, an attempt was made to assign a new value to a numeric value. The only entities which can be assigned a new value are attributes, Variables, and local variables.

ERROR - ******** GLOBAL CONSTANT CANNOT BE ASSIGNED A NEW VALUE

In the definition of an ASSIGN Primitive in a Process, an attempt was made to assign a new value to a global Constant. The only entities which can be assigned a new value are attributes, Variables, and local variables.

ERROR - '********' - NOT RECOGNIZED AS A LOGICAL RELATION

In the definition of a COMPARE Primitive in a Process, the relation field was invalid. Valid relations are EQ, NE, GE, GT, LE, and LT.

ERROR - '********' IS NOT RECOGNIZED AS AN ARITHMETIC OPERATION OR A LOCAL VARIABLE

In the definition of an EVAL Primitive in a Process, the function specified was invalid. The function field can also contain the name of a local variable which is a reference to a defined Table.

ERROR - '********' A GLOBAL CONSTANT NUMERIC OR KEYWORD CANNOT BE ASSIGNED TO

In the definition of an EVAL Primitive in a Process, a global Constant, numeric or a keyword was specified in the set variable

field. The only entities which can be assigned a new value by an EVAL are global Variables and local variables.

ERROR - ******** NUMERIC REFERENCE INVALID IN PROCESS FIELD

In the definition of a SEND Primitive in a Process, the Process field contained a numeric reference. The Process field must contain a reference of a defined Process.

ERROR - ******** REFERENCE INVALID IN ITEM FIELD

In the definition of a SEND Primitive in a Process, the list of Items to be sent to a Process contained an invalid value. Only Items can be sent to a Process.

In the definition of a CREATE Primitive in a Process, the list of Items to be created included an invalid value. Only Items can be created by a CREATE Primitive.

In the definition of a DESTROY Primitive in a Process, the list of Items to be destroyed included an invalid value. Only Items can be destroyed by a DESTROY Primitive.

In the definition of a FILE Primitive in a Process, the Item field contained an invalid value. The Item field must contain the name of a reference to a defined Item.

In the definition of a FIND Primitive in a Process, the Item field contained an invalid value. The Item field must contain the name of a local variable to be set.

In the definition of a REMOVE Primitive in a Process, the Item field contained an invalid value. The Item field must contain the name of a variable to be set.

ERROR - ******** INVALID QUEUE OPTION

In the definition of a FILE Primitive in a Process, the option field contained an invalid option. The valid options are FIRST, LAST, NEXT, and BEFORE.

In the definition of a FIND Primitive in a Process, the option field contained an invalid option. The valid options are FIRST, LAST, NEXT, and BEFORE.

In the definition of a REMOVE Primitive in a Process, the option field contained an invalid option. The valid options are FIRST, LAST, and NEXT.

ERROR - ******** REFERENCE INVALID IN QUEUE FIELD

In the definition of a FILE Primitive in a Process, the queue field contained an invalid value. The queue field must contain the name of a reference to a defined Queue.

In the definition of a FIND Primitive in a Process, the queue field contained an invalid value. The queue field must contain the name of a reference to a defined Queue, or the name of a valid cross-reference set: Action, Constant, Item, Process, Queue, Resource, Table, or Variable.

In the definition of a REMOVE Primitive in a Process, the queue field contained an invalid value. The queue field must contain the name of a reference to a defined Queue.

ERROR - ******** - RESUME REFERENCE MUST NOT BE NUMERIC OR GLOBAL

In the definition of the RESUME Primitive, a numeric value or a Constant or Variable was encountered in the Process field. This reference must be a local variable.

- ERROR - TRACE MODE MUST BE EITHER 'ON' OR 'OFF'

In the definition of a TRACE Primitive, the ON/OFF field contained a value other than "ON" or "OFF". These are the only valid values.

ERROR - LOAD NODE IS NOT RECOGNIZED AS A RESOURCE

In the definition of a Load entity, a value was encountered in a node field which was not a reference to a defined Resource. Nodes must be Resources.

ERROR - '********' IS NOT DEFINED AS A PROCESS

In the definition of a Load, the name specified in the process field was not defined as a Process. The name specified in this field must be a defined Process.

ERROR - ******** IS NOT A LOAD DISTRIBUTION FUNCTION

In the definition of a LOAD, the name specified in the schedule field was not a valid Load distribution.

ERROR - ******** IS NOT DEFINED AS A CONSTANT OR VARIABLE

In the definition of a Load, a non-numeric value in the rate field was not defined as a global Constant or variable. If the rate field contains a non-numeric value, it must be a defined global Constant or Variable.

In the definition of a Load, a non-numeric value in the mean field was not defined as a global Constant or Variable. If the mean field contains a non-numeric value, it must be a defined global Constant or Variable.

In the definition of a Load, a non-numeric value in the delta field was not defined as a global Constant or Variable. If the delta field contains a non-numeric value, it must be a defined global Constant or Variable.

In the definition of a Load, a non-numeric value in the priority field was not defined as a global Constant or Variable. If the priority field contains a non-numeric value, it must be a defined global Constant or Variable.

ERROR - NO SCENARIO DEFINED

No Scenario was defined. There must be a Scenario defined in order to run a simulation on a model.

ERROR - PERIOD NOT DEFINED

In the definition of a Scenario, the period was not defined. The period length for a Scenario can be a numeric value or a defined Constant.

ERROR - TRIGGER ******** NOT DEFINED AS A LOAD OR PROCESS

In the definition of a Scenario entity, a value in the trigger field was not a Load or Process. Scenario triggers must be either Loads or Processes.

WARNING - '********' DISTRIBUTION ONLY REQUIRES 1 PARAMETER

In the definition of an ACTION Primitive in a Process, the specified distribution required only one parameter, but two were supplied. The extra parameter should be deleted or the distribution should be changed.

WARNING - **** NOT LEGAL. USING D INSTEAD.

An illegal Table type was specified. The Table is being assumed to be discrete. The valid table types are continuous (c), discrete (d), and alpha (a).

WARNING - ATTRIBUTE INITIAL VALUE IS NOT DEFINED

In the definition of an Item, Process, or Resource an attribute was not assigned an initial value or was assigned an invalid value. Attributes must be initialized.

WARNING - BLANK PRIORITY FIELD ASSUMES PRIORITY 0

In the definition of a CALL Primitive of a Process, the priority field was left blank. The priority is assumed to be zero.

In the definition of a LOAD entity, the priority field was left blank. The priority is assumed to be zero.

In the definition of a Scenario entity, the priority field was left blank. The priority is assumed to be zero.

WARNING - ******** IS AN ILLEGAL OPTION. USING NOWAIT INSTEAD.

In the definition of a CALL Primitive of a Process, the option field contained an invalid option; a NOWAIT option is being assumed. The valid options are BLOCK, WAIT, and NOWAIT.

WARNING - '********' IS NOT RECOGNIZED IN THIS CONTEXT

In the definition of an ASSIGN Primitive of a Process, an attempt was made to assign a numeric value or a Constant or Variable, but there was also a value in the qualifier field. The qualifier is being ignored.

In the definition of an ASSIGN Primitive in a Process, an attempt was made to assign a value to the $CNODE keyword, or an attempt was made to assign a value to a Variable, but there was also a value in the qualifier field. The qualifier is being ignored.

WARNING - '********' - NO QUALIFICATION RECOGNIZED FOR IDENTIFICATION

In the definition of a COMPARE Primitive in a Process, a unrecognizable qualifier for a numeric, a global Variable, or a global Constant was encountered. Qualifiers are allowed only for Items, Processes, Resources, and certain keywords.

WARNING - ******** IS NOT RECOGNIZED IN THIS CONTEXT FOR FUNCTION

In the definition of an EVAL Primitive in a Process, operands were specified with a random function or a second operand was specified for a function which only required one operand.

WARNING - ******** IS NOT AN ACTION DISTRIBUTION - USING CONSTANT

In the definition of an ACTION primitive in a Process, the value in the method field was not a valid Action distribution; the distribution is being assumed to be CONSTANT. The valid distributions are exponent, constant, lognormal, normal, uniform, Weibull, gamma, and Erlang.

If an execution error occurs during the simulation, execution will halt and an error message will be printed in the statistical summary. In some cases there may be a Simscript II.5 traceback. This traceback is a hexadecimal formatted report which is to be disregarded by the user. Following the error messages, the statistical summary lists the state of the Process which was executing when the error occurred. The value of all local variables and attached attributes for the Process are listed. All other output reports are also generated.

Following are all of the execution errors which are produced and an explanation of the conditions which cause each error.

EXECUTION ERROR DETECTED IN PROCESS ********

An error occurred in the specified Process which caused an abnormal termination of the simulation.

EXECUTION ERROR - BRANCH PROBABILITY FOR CURRENT STATEMENT IS NOT A NUMBER

The BRANCH probability in a BRANCH Primitive in a Process does not evaluate to a number.

EXECUTION ERROR - LOOP NUMBER FOR CURRENT STATEMENT IS NOT A NUMBER

The value of the LOOP counter in a Process is not a number.

EXECUTION ERROR - TEST STATEMENT ENTITY IS NOT A RESOURCE OR QUEUE

The value to be tested by a TEST Primitive in a Process is not a Resource or a Queue. The TEST Primitive can only test a Resource or a Queue.

EXECUTION ERROR - VALUE OF RESET IN CURRENT STATEMENT IS NOT A NUMBER

The value for the number of units to be reset by a RESET Primitive is not a number. The value for the number of units to be reset must evaluate to a number.

EXECUTION ERROR - ATTEMPT TO RESET # OF RESOURCE UNITS OUTSIDE OF LEGAL LIMITS

An attempt was made to reset a number of Resource units which would make the number of units inactive or active greater than the total number of units which were defined for this Resource.

EXECUTION ERROR - VALUE OF UNITS REQUESTED IN CURRENT STATEMENT IS NOT A NUMBER

The units field in an ALLOC Primitive did not resolve to a number. This field must resolve to a number.

EXECUTION ERROR - VALUE OF PRIORITY IS NOT LEGAL

The Priority field in an ALLOC Primitive did not resolve to a number. This field must resolve to a number.

EXECUTION ERROR - VALUE OF UNITS TO BE RELEASED IN CURRENT STATEMENT IS NOT A NUMBER

The units field in a DEALLOC Primitive did not resolve to a number. This field must resolve to a number.

EXECUTION ERROR - RESUME ATTEMPTS TO RESUME A PROCESS WHICH IS NOT SUSPENDED

An attempt was made to resume a Process instance which was not suspended.

EXECUTION ERROR - A REFERENCE IN THE CURRENT PROCESS EVALUATES TO AN ILLEGAL TYPE FOR THE CURRENT STATEMENT

The Resource field in a RESET Primitive did not resolve to a Resource. This field must resolve to a defined Resource entity.

The Resource field in an ALLOC Primitive did not resolve to a Resource. This field must resolve to a defined Resource entity.

The Resource field in a DEALLOC Primitive did not resolve to a Resource. This field must resolve to a defined Resource entity.

EXECUTION ERROR - AN ACTION REFERENCE DOES NOT EVALUATE TO A NUMBER

The scheduling time or the scheduling delta time for an action does not evaluate to a number.

EXECUTION ERROR PRIMITIVE REFERENCE DOES NOT EVALUATE TO AN ACTION

An undefined opcode for a Primitive was encountered. The opcode was assumed to be the name of a reference to an Action, but it did not resolve to a defined Action name.

EXECUTION ERROR - PROCESS IN CURRENT CALL STATEMENT IS NOT DEFINED AS A PROCESS

An attempt was made by a CALL Primitive to initiate a Process which was not defined. The Process name in a CALL Primitive must be a reference to an entity defined as a Process.

EXECUTION ERROR - PRIORITY IN CALL DOES NOT EVALUATE TO A NUMBER

The priority in a CALL Primitive did not evaluate to a number. The priority for calling a Process must evaluate to a number.

EXECUTION ERROR - DISAGREEMENT IN NUMBER OF GIVEN PARAMETERS BETWEEN CURRENT CALL STMT AND CALLED PROCESS

The number of given parameters in a CALL Primitive differs from the number of given parameters in the definition of the Process to be called. These parameters must correspond.

EXECUTION ERROR - DISAGREEMENT IN NUMBER OF RETURN PARAMETERS BETWEEN CURRENT CALL STMT AND CALLED PROCESS

The number of return parameters in a CALL Primitive differs from the number of return parameters in the definition of the Process to be called. These parameters must correspond.

EXECUTION ERROR - ORDER RELATIONS ARE NOT DEFINED FOR COMPARE TYPES

For the non-numeric types being compared, an invalid relation was specified. The only valid relations for these types is equal or not equal.

EXECUTION ERROR - EVAL VARIABLE DOES NOT EVALUATE TO A NUMBER

One of the variables in an EVAL Primitive for a function other than a Table does not evaluate to a number.

EXECUTION ERROR - EVAL FUNCTION IS NOT RECOGNIZED AS AN ARITHMETIC OPERATOR OR A TABLE REFERENCE

The reference for the function in an EVAL Primitive is not a legal arithmetic function or a reference to a defined Table.

EXECUTION ERROR - EVAL VARIABLE FOR DISCREET OR CONTINUOUS TABLE DOES NOT EVALUATE TO A NUMBER

In an EVAL Primitive which is being used to look up a value in a Table, the value used to index into the Table, the x value, does not evaluate to a number.

EXECUTION ERROR - ILLEGAL ASSIGN: CURRENT NODE MUST BE SET TO A RESOURCE

An EVAL Primitive attempted to set the current node to a reference which was not a defined Resource. The current node must be a Resource.

EXECUTION ERROR - ASSIGN ATTEMPTS TO MODIFY A QUALIFIED TYPE FOR WHICH NO ATTRIBUTE IS DEFINED

An attempt was made to assign a new value to an attribute of an entity for which no attributes can be defined. Only Processes, Resources, and created Items have attributes which can be modified.

EXECUTION ERROR - ******* ATTRIBUTE NOT DEFINED FOR ITEM

An attempt was made to assign a new value to a nonexistent attribute of an Item.

EXECUTION ERROR - ******* ATTRIBUTE NOT DEFINED

An attempt was made to assign a new value to a nonexistent attribute of a Process or a Resource.

EXECUTION ERROR - ASSIGN ATTEMPTS TO MODIFY A TYPE WHICH CANNOT BE MODIFIED

An attempt was made to assign a new value to an entity which cannot be modified; i.e., a global Constant, a number, or a keyword other than $CNODE.

EXECUTION ERROR - ATTEMPT TO CREATE AN ENTITY WHICH IS NOT AN ITEM

An attempt was made to create an entity which is not an Item. Only references to Items may be in the create list of the CREATE Primitive.

EXECUTION ERROR - ATTEMPT TO DESTROY AN ITEM WHICH IS CURRENTLY FILED ON A QUEUE

An attempt was made to destroy an Item which had been filed on a Queue and not removed before execution of the DESTROY Primitive.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTS TO DESTROY AN ITEM WHICH IS NOT DEFINED OR DOES NOT EXIST

An attempt was made to destroy an Item which was not defined or created, or which has already been destroyed.

EXECUTION ERROR - PROCESS FIELD IN SEND STATEMENT IS NOT DEFINED AS A PROCESS

The reference in the Process field of a SEND Primitive was not resolved as a Process. Items can only be sent to a defined Process.

EXECUTION ERROR - ATTEMPT TO SEND AN ITEM WHICH IS CURRENTLY FILED ON A QUEUE

An attempt was made to send an Item which is currently filed on a Queue to another Process before the Item was removed from the Queue.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTING TO SEND A ENTITY WHICH IS NOT DEFINED AS AN ITEM

An attempt was made by a SEND Primitive to send an entity other than an Item to a Process. Only references to Items may be specified in the SEND Primitive to be sent to Processes.

EXECUTION ERROR - ITEM ******** ATTEMPT TO BE RECEIVED BY PROCESS ******** IS NOT IN PROCESS NEED LIST

An attempt was made to cause a Process to receive an Item which was not on the list of Items which the Process should receive.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTS TO FILE AN ENTITY WHICH CANNOT BE FILED

An attempt was made by a FILE Primitive to file an entity which cannot be filed. Only Items can be filed.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTS TO FILE AN ITEM ON AN UNDEFINED QUEUE

An attempt was made to FILE an Item on a Queue which was not defined. The queue reference in the FILE primitive must resolve to a defined Queue.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTS TO FILE AN ITEM WHICH IS ALREADY ON A QUEUE

An attempt was made to refile an Item. An Item can be filed on only one Queue at any given time.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTS TO FILE AN ENTITY BEFORE OR AFTER AN UNDEFINED ENTITY

An attempt was made to file an entity before or after an entity which did not exist on the Queue.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTING TO REMOVE AN ITEM FROM AN UNDEFINED QUEUE

An attempt was made to remove an entity from an undefined Queue.

EXECUTION ERROR - CURRENT PROCESS ATTEMPTING TO REMOVE 'NEXT' ITEM WHICH DOES NOT EXIST

An attempt was made to remove a non-existent current Item from a Queue.

EXECUTION ERROR - A REFERENCE FOR A QUEUE IN A FIND PRIMITIVE IS NOT DEFINED AS A QUEUE OR XREF SET

An invalid reference was specified in a FILE Primitive as the queue name. Only Queues or cross reference sets are valid for the Queue name field.

EXECUTION ERROR - ******* ATTRIBUTE OF A RESOURCE IS NOT DEFINED

An attempt was made to reference a non-existent attribute of a Resource. Valid attributes are NIDLEQ, NBUSYQ, NWAITQ, NINACTQ, as well as user-modifiable attributes.

EXECUTION ERROR - ******* ATTRIBUTE OF A RESOURCE UNIT NOT DEFINED

An attempt was made to reference a non-existent attribute of a Resource unit. Valid attributes are NIDLEQ, NBUSYQ, NWAITQ, NINACTQ, as well as user-modifiable attributes.

EXECUTION ERROR - ******** ATTRIBUTE OF A PROCESS IS NOT DEFINED

An attempt was made to reference a non-existent attribute of a Process.

EXECUTION ERROR - ******* ATTRIBUTE OF A TASK IS NOT DEFINED

An attempt was made to reference a non-existent attribute of a Task.

EXECUTION ERROR - ******* ATTRIBUTE OF A QUEUE NOT DEFINED

An attempt was made to reference an invalid attribute of a Queue. The valid attributes are NQUEUE and TQUEUE.

EXECUTION ERROR - ******* ATTRIBUTE IS NOT DEFINED FOR CURRENT ITEM REFERENCE ******* IN EXECUTING LOGIC

An attempt was made to reference a non-existent attribute of an Item.

EXECUTION ERROR - ******* ATTRIBUTE SPECIFIED FOR A TYPE FOR WHICH NO ATTRIBUTES CAN BE DEFINED

An attempt was made to reference an attribute of a type which does not have attributes. Entities which have attributes are Resources, Processes, and Items.

EXECUTION ERROR - KEYWORD REFERENCE IS BLANK

When the simulator tried to resolve a Keyword, the reference field for the parameter was found to be blank.

EXECUTION ERROR - PROCESS NODE HAS NOT BEEN DEFINED

An attempt was made to reference the process node of a Process, but the node was not defined.

EXECUTION ERROR - REFERENCE FOR $ NODE IS NOT A PROCESS

When the simulator tried to resolve the Keyword $NODE, the reference was not a Process.

EXECUTION ERROR - ROUTE SET ERROR - NO PATH IN NETWORK

When the simulator tried to resolve $NXTNODE or $LINK, there was no valid path defined in the LPT.

EXECUTION ERROR - CNODE FOR EXECUTING PROCESS NOT DEFINED

When the simulator tried to resolve a link or a next node, there was no current node defined for the executing Process.

ERROR - ILLEGAL TYPE

In a Process which was executing when an abnormal termination of the simulation occurred, a local variable was of an invalid type.

ERROR - ILLEGAL ATTRIBUTE TYPE

In a Process which was executing when an abnormal termination of the simulation occurred, a local variable which resolved to an Item, a Process, or a Resource had an invalid attribute.

WARNING - ALPHA TABLE LOOKUP FAILED

In an EVAL Primitive being used to look up a value in an alpha Table, a value was not found which corresponded to the lookup index value.

WARNING - EMPTY TABLE DETECTED

A Table was encountered which did not have any entries in it.

SIMULATOR ERROR IN COMPUTING NEXT TIME ON
METHOD = ******** REF1 = ******** REF2 = ******** STREAM = ********

An error occurred when an attempt was made to compute the next time in the simulation.

INTERNAL SIM ERROR TRYING TO SUSPEND MORE UNITS OF
******** THEN

An error occurred when an attempt was made to deallocate units of a suspended Process instance.

SIMULATOR ERROR - QUEUE.START ATTEMPT TO FILE UNSUCCESSFUL

The simulator attempted to restart tasks blocked from a Queue when there were no tasks currenty blocked.

APPENDIX C

GLOSSARY

ACTION - A discrete event that consumes time during a simulation run.

ANALYSIS USER INTERFACE (AUI) - The interface between the user and the AISIM simulator.

ANALYSIS USER INTERFACE (AUI) READY STATE - Any time after the Analysis User Interface has been invoked, except during a simulation period. This state is indicated by the "*" prompt.

ARCHITECTURE DESIGN EDITOR (ADE) - A sublevel of the DUI which provides the user with the graphics commands to construct a system architecture.

ARCHITECTURE DESIGN EDITOR (ADE) MENU - A representation of the valid symbols available to the user during an ADE session for building an architecture. See ADE MENU.

ARCHITECTURE DESIGN EDITOR (ADE) READY STATE - The state of the system while in the ADE that allows the user to enter commands. This state is indicated by the "#" prompt.

ATTRIBUTE - The specific characteristic of a defined entity.

ATTRIBUTE FORM - A list of available attributes from which the user must select one attribute to be used for testing or data sampling.

BLOCK - Used in conjunction with the CALL Primitive (see section 3.9.5) to indicate that the calling task is to call the specified task and wait until all associated tasks are complete before continuing.

BREAKPOINT - A user-specified condition which, when reached, suspends the simulation to allow the user to monitor the current state of the simulation.

CONSTANT - A value that is not subject to change once a simulation run has been started.

DATABASE - The accumulation of data in a specified form related to a specific function or operation.

DEFAULT CONDITION - The condition that exists if no parameters are explicitly stated.

DESIGN USER INTERFACE (DUI) - The interface that allows the user to create or modify a design database.

DESIGN USER INTERFACE (DUI) READY STATE - Any time after invocation of the Design User Interface, except when utilizing the PEI or ADE sublevels of the DUI. This state is indicated by the "*" prompt.

ENTITY - A predefined set of constructs that have user defined attributes (see section 3 for valid AISIM entities). They are the "building blocks" with which the user creates his model.

ENTITY-NAME - The user-defined name of a valid entity.

ENTITY-TYPE - A type as opposed to a specific, user-defined instance of an entity.

FORMS MODE - A specific function that provides areas which may be filled in by the user, and protected fields which define the areas to be filled in.

INFINITE RESOURCES - A feature which allows the simulator to simulate a Process as if there were no limit to the number of Resources available to it.

LOAD - The amount of activity to be applied to the simulation of a process.

L-NODE - A leaf node in an architecture which typically represents an external load on the system.

MODEL - A group of AISIM entities which represent a certain function or group of functions.

NOWAIT - Used in conjunction with the CALL Primitive to indicate that a Process is to be called by a parent Process and the parent Process is to continue processing in parallel.

OFF-SCREEN - The portion of a graphics picture not visible to the user.

ON-SCREEN - The portion of a graphics picture visible to the user.

PERMANENT DATABASE (sometimes referred to as the Design database) - The user-named database, in which the data for a modeled system resides. (As opposed to the working database which temporarily holds Design data while editing that data).

PRIMITIVE - The model entity used to model individual steps in an operation or function. A Process is constructed from a sequence of Primitives.

PROCESS - A graphical representation of a sequence of events, activities and decisions that models a real-world operation or function.

PROCESS EDITOR INTERFACE (PEI) - A sublevel of the DUI that provides the user with the graphics commands to construct Processes.

APPENDIX D

MESSAGE ROUTING SUBMODEL

The message routing submodel provides a means for a user to route messages through a network which is defined by an architecture and a Legal Path Table.

The message routing submodel consists of one Item representing the message dispatched through the system architecture, four Processes representing the activities required for the inter-node communication and other supporting entities. Everything required for this model is included in the AISIM system library and can be merged into a user's model in a simple operation. (See the Library User Interface, section 10).

Although intra-node communication is modeled by means of a collection of four Processes, the user need explicitly invoke with a CALL Primitive (see section 3.9.5) only one of them. To represent the intra-Node triggering of a Process one calls the first Process in the submodel called "MRS". This process is called using a WAIT option if the user wishes to suspend the calling Process until message routing submodel processing is complete. It allows the calling Process to wait for a response message to be sent back to the calling node before it continues processing. If the MRS Process is called with a NOWAIT option, processing in the message routing submodel will proceed concurrently with the calling Process.

The calling process must call process MRS with six GIVEN values; no RETURN values are required. The GIVEN parameters are:

1. the name of the destination process to be triggered,

2. the priority associated with the destination process,

3. the type of message to be generated -- $REQRESP, $REQNORE, or $RESP. $REQRESP causes a response message to be sent to the origin, $REQNORE causes no response message to be generated, and $RESP inhibits both the response message and the triggering of a destination process,

4. the length of the message,

5. the destination node, and

6. the name of the message item.

The user must also set up attributes of the Resources representing the nodes and channels (see section 3.6). All nodes which messages utilize

must have an M.ROUTE attribute which gives the nodal processing delay in time units per message. Each channel resource must have a RATE attribute giving a channel transmission delay in time units per character.

The entities that comprise the message routing submodel are described in the following sections.

Item MSG

This Item is the basic prototype for messages created by the MRS. If the user does not want specific point-to-point transit times, all statistics for message routing will be accumulated for this one entity. If specific point-to-point transit times are desired, the AISIM user copies MSG to another Item name (through the Design User Interface COPY command described in section 6.1.2) and provides the unique name as parameter six in the call to the MRS Process. All attributes of the Item MSG are essential to the message routing submodel. The attributes are explained below.

DEFINITION OF ATTRIBUTES FOR ITEM MSG:

Attribute Name	Default Value	Description
CNODE	$CNODE	The current node where the message resides.
FNODE	$CNODE	The source node of the message.
LENGTH	99999999	The length of the message in bytes.
TYPE	$REQNORE	The message type. $REQRESP is a request message requiring a response when the destination is reached. $REQNORE is a request message with no response required. $RESP is a response message.
RPROC	$ERROR	The destination Process name.
RPROCPRI	99999999	The priority for the destination Process.
TNODE	$CNODE	The destination node.

Resources

No Resources are contained in the message routing submodel. However, to use it, the user must specify an architecture. Each nodal Resource which messages utilize must have an M.ROUTE attribute which gives the nodal processing delay in time units per message. Each channel Resource must have a RATE attribute giving a channel transmission delay in time units per character.

Actions

Two Action entities are used by the message routing submodel -- ROUTE.OH and XFER.OH. ROUTE.OH is found in Process NODEPROC and is used for nodal processing delays while XFER.OH is found in Process CHANPROC for channel transmission delays.

MESSAGE ROUTING SUBMODEL PROCESSES

The message routing submodel contains four Processes. Details of these Processes do not have to be known by the user if the submodel can be used as is. However, if the user needs to make changes, knowledge of how the processes work is essential. This subsection describes the functioning of these Processes.

Process MRS

This is the top-level Process of the message routing submodel and is called when a user wishes to make use of the submodel. It causes a request message to be generated. Following is a list of the parameters of this Process.

PROCESS NAME: MRS — Generate a request message and pass it to NODEPROC.

LOCATION: Executes in all nodes.

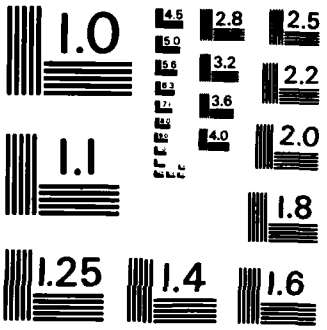
GIVEN: PROCESS (DATA TYPE: PROCESS) -- the name of the process to be initiated in the destination node.

PRIORITY (DATA TYPE: REAL) -- the priority of the destination process.

MSG.TYPE (DATA TYPE: ALPHA) -- the type of message to be created. The only legal values for this parameter are: $REQNORE -- a request message is created which requires no response, $REQRESP -- a request message is created which will request a response at its destination, $RESP — used only if no process is to be initiated in the destination node and no response is required.

MSG.LNTH (DATA TYPE: REAL) -- the message length.

AD-A161 556 ASIM (AUTOMATED INTERACTIVE SIMULATION MODELING SYSTEM) 4/4
VAX VERSION USER'.. (U) HUGHES AIRCRAFT CO FULLERTON CA
GROUND SYSTEMS GROUP S KNEEBURG FEB 85 ESD-TR-85-127
UNCLASSIFIED F33615-81-C-5098 F/G 9/2 NL



TO.NODE (DATA TYPE: RESOURCE) -- the message destination node.

MSG (DATA TYPE: ITEM) -- the name of the message item to be created.

RETURN: None

CALLS: NODEPROC

The Process begins by creating a message and initializing various attributes of it. The attributes CNODE and FNODE are initialized to the current node in which the Process is executing. The attribute RPROC is set to the Process that will be triggered at the message destination node. The attribute RPROCPRI is set to the priority at which the requested Process will execute. The attribute TYPE is set to the parameter passed in MSG.TYPE. The attribute length is set to the length of the message. The destination node is stored in the TNODE attribute. Process MRS then calls Process NODEPROC with a WAIT option and gives it the newly created message. Figure D-1 is a listing of this Process.

```
PROCESS
MNEMONIC          DESCRIPTION
========= ========= ================================================
MRS               GENERATE A PROCESS REQUEST MESSAGE AND INITIATE I/O


ENTRY     OPCODE    PARM      PARM      PARM      COMMENT
========= ========= ========= ========= ========= ========================
          START     ALL
          GIVEN     PROCESS   PRIORITY  MSG.TYPE
                    MSG.LNTH  TO.NODE   MSG
          CREATE    MSG                         CREATE MESSAGE
          ASSIGN    MSG.LNTH                    SET MESSAGE LENGTH
                    MSG       LENGTH
          ASSIGN    PROCESS                     SET PROCESS
                    MSG       RPROC
          ASSIGN    PRIORITY                    SET PRIORITY
                    MSG       RPROCPRI
          ASSIGN    TO.NODE                     SET DESTITION
                    MSG       TNODE
          ASSIGN    MSG.TYPE                    SET MESSAGE TYPE
                    MSG       TYPE
          CALL      NODEPROC  WAIT      PRIORITY EXECUTIVE SERVICING OF MSG
          GIVEN     MSG
          END

LOCAL VARIABLES OF PROCESS MRS
================================================================================
    1 PROCESS    (X)     2 PRIORITY           3 MSG.TYPE          4 MSG.LNTH
    5 TO.NODE            6 MSG         (I)    7 NODEPROC   (P)
```

Figure D-1. Listing of Process MRS

Process NODEPROC

This Process performs nodal processing and determines whether the message is at its destination. When the Process is called, it is given the message Item. The following describes the parameters of the Process.

PROCESS NAME:	NODEPROC -- Nodal Processing
LOCATION:	Executes in all nodes.
GIVEN:	MSG (DATA TYPE: ITEM) -- This parameter is the name of the message item created in process MRS.
RETURN:	None
CALLS:	CHANPROC, DESTPROC

The first step of this Process is to assign the name of the current node to a system variable. The processing delay is then calculated and charged against the current node.

The message's current position is compared with its destination. If the message is at its destination, the Process determines whether the message is a request or response message. If it is a request message, the Process DESTPROC is called with a WAIT option and a priority equal to the requested priority. The requested Process is initiated in the destination node by the Process DESTPROC. If the message is a response message, the Process DESTPROC is called to destroy the message. If the message is not at its destination node, the Process CHANPROC is called to forward the message to its next node. Figure D-2 is a listing of this Process.

```
PROCESS
MNEMONIC          DESCRIPTION
========= ======== ==================================================
NODEPROC          NODAL PROCESSING AND ROUTING


ENTRY     OPCODE   PARM     PARM     PARM     COMMENT
========= ======== ======== ======== ======== ========================
          START    ALL
          GIVEN    MSG
          ASSIGN   MSG      CNODE             INDICATE CURRENT NODE
                   C.NODE
          ASSIGN   C.NODE   M.ROUTE           PROCESSING RATE OF NODE
                   RT.OVHD
          ASSIGN   MSG      LENGTH            GET MESSAGE LENGTH
                   MSG.LNTH
          EVAL     OVERHEAD MULTIPLY          COMPUTE PROCESSING DELAY
                   MSG.LNTH RT.OVHD
          ALLOC    C.NODE   1        ALL      ALLOCATE CURRENT NODE
                   $PRIORTY
          ROUTE.OH CONSTANT OVERHEAD          DELAY FOR ROUTING
          DEALLOC  C.NODE   1                 RELEASE C.NODE TO OTHERS
          COMPARE  MSG      CNODE    EQ       IS MSG AT DESTINATION?
                   MSG      TNODE    CONTROL
          CALL     CHANPROC WAIT     0        FORWARD MSG TO CHANNEL
          GIVEN    MSG
          BRANCH   END      100
CONTROL   ENTRY                               MESSAGE AT DESTINATION
          CALL     DESTPROC WAIT     0        CONTEXT SWITCH MESSAGE
          GIVEN    MSG
END       ENTRY
          END

LOCAL VARIABLES OF PROCESS NODEPROC
=====================================================================
    1 MSG       (I)     2 C.NODE          3 MSG.LNTH          4 OVERHEAD  (A)
    5 ROUTE.OH  (A)     6 CHANPROC  (P)     7 DESTPROC  (P)
```

Figure D-2. Listing of Process NODEPROC

Process DESTPROC

This Process models the processing of a message at its destination. It terminates request messages, generates response messages and triggers the requested Process. The following describes the parameters of this Process.

PROCESS NAME: DESTPROC -- Destination processing of message items.

LOCATION: Executes in all nodes.

GIVEN: MSG (DATA TYPE: ITEM) - This parameter is the message item created in process MRS.

RETURNS: None

CALLS: CHANPROC

This Process determines whether the message is a request or response message. If it is a response message, this Process destroys the message and terminates. If the message is a request message, the name of the requested Process is retrieved from the RPROC attribute of the message, and the process is initiated. DESTPROC waits until the requested process completes. Next, DESTPROC checks the message attribute TYPE to see whether the requesting Process is waiting for a response. If no response is desired, the message is destroyed and DESTPROC terminates. If a response is requested, the message type is changed to response, the destination node is changed to the origin, and the origin is changed to the current node. Then the Process CHANPROC is called to route the message back to its origin. Figure D-3 is a listing of this Process.

```
PROCESS
MNEMONIC          DESCRIPTION
========= ========= ==================================================
DESTPROC          PROCESSING AT DESTINATION OF MESSAGE

ENTRY     OPCODE    PARM      PARM      PARM      COMMENT
========= ========= ========= ========= ========= ========================
          START     ALL
          GIVEN     MSG
          ASSIGN    MSG       CNODE               CURRENT NODE
                    C.NODE
          COMPARE   MSG       TYPE      EQ        IF RESPONSE, DESTROY
                    $RESP               DESTROY
          ALLOC     C.NODE    1         ALL       ALLOCATE CURRENT NODE
                    $PRIORTY
          ASSIGN    MSG       RPROC               EXECUTE THE CALLED PROCESS
                    PROCESS
          ASSIGN    MSG       RPROCPRI            SET PRIORITY FOR REQ PROC
                    PRIORITY
          CALL      PROCESS   WAIT      PRIORITY  WAIT UNTIL COMPLETE
          GIVEN     MSG
          RETURN    MSG
          DEALLOC   C.NODE    1                   DEALLOCATE CURRENT NODE
          COMPARE   MSG       TYPE      EQ        NO RESPONSE REQ->DESTROY
                    $REQNORE            DESTROY
          ASSIGN    $RESP                         CHANGE MSG RESPONSE TYPE
                    MSG       TYPE
          ASSIGN    MSG       FNODE               SWITCH FROM AND TO NODES
                    MSG       TNODE
          ASSIGN    MSG       CNODE               CURRENT NODE IS FROM NODE
                    MSG       FNODE
          CALL      CHANPROC  WAIT      0         RETURN MESSAGE TO ORIGIN
          GIVEN     MSG
          BRANCH    END       100
DESTROY   ENTRY                                   TERMINATE MESSAGE AT DEST.
          DESTROY   MSG                           TERMINATE MSG
END       ENTRY
          END

LOCAL VARIABLES OF PROCESS DESTPROC
=============================================================================
    1 MSG       (I)     2 C.NODE          3 PROCESS   (X)     4 PRIORITY
    5 CHANPROC  (P)
```

Figure D-3. Listing of Process DESTPROC

Process CHANPROC

Process CHANPROC extracts the current node and the destination node from the message Item. It accesses the LPT to determine the next node and the connecting channel. The channel is allocated to simulate its use, and the Process NODEPROC is called. The following describes the parameters of process CHANPROC.

PROCESS NAME: CHANPROC -- full and half duplex channel logic

LOCATION: Executes in all nodes

GIVEN: MSG (DATA TYPE: ITEM) -- This parameter is the message item created in MRS.

RETURN: None

CALLS: NODEPROC

The first step of this Process is to assign the current node to the system variable $CNODE and to determine the destination node for the message. Then the next node and next channel are extracted from the LPT, and the channel is allocated. The transfer time for the message is assumed to be a constant rate. The action XFER.OH simulates the time used to traverse the channel. The value of the current-node attribute of the message is changed to the next node to update the message's position, and the system current node indicator ($CNODE) is also set to the next node. The channel is then deallocated and the Process NODEPROC is called. Figure D-4 is a listing of this Process.

```
PROCESS
MNEMONIC          DESCRIPTION
========= ========= ==================================================
CHANPROC          FULL AND HALF DUPLEX CHANNEL LOGIC

ENTRY     OPCODE    PARM      PARM      PARM      COMMENT
========= ========= ========= ========= ========= ========================
          START     ALL
          GIVEN     MSG
          ASSIGN    MSG       CNODE               SET INTERNAL NODE CURRENT
                    $CNODE
          ASSIGN    MSG       TNODE               GET DESTINATION NODE
                    TO.NODE
          ASSIGN    $NXTNODE  TO.NODE             SET NEXT NODE TO DESTN
                    NXT.NODE
          ASSIGN    $CHANNEL  TO.NODE             GET CHANNEL TO NEXT NODE
                    CHANNEL
          ALLOC     CHANNEL   1         ALL       OBTAIN CHANNEL FOR XFER
                    $PRIORTY
          ASSIGN    CHANNEL   RATE                WHAT IS CHANNEL RATE?
                    VSPEED
          ASSIGN    MSG       LENGTH              MESSAGE LENGTH
                    VLENGTH
          EVAL      VM.OVHD   MULTIPLY            CALCULATE TRANSFER TIME
                    VSPEED    VLENGTH
          XFER.OH   CONSTANT  VM.OVHD             DELAY DUE TO TRANSFER TIME
          ASSIGN    NXT.NODE                      MSG RESIDES IN NEXT NODE
                    MSG       CNODE
          ASSIGN    NXT.NODE                      SET INTERNAL NODE REGISTER
                    $CNODE
          DEALLOC   CHANNEL   1                   FREE UP CHANNEL AFTER XFER
          CALL      NODEPROC  WAIT      0         ROUTE MESSAGE TO NEXT NODE
          GIVEN     MSG
          END

LOCAL VARIABLES OF PROCESS CHANPROC
=============================================================================
    1 MSG        (I)     2 TO.NODE          3 NXT.NODE          4 CHANNEL
    5 VLENGTH            6 VM.OVHD          7 XFER.OH    (A)    8 NODEPROC   (P)
```

Figure D-4. Listing of Process CHANPROC

END

FILMED

1-86

DTIC